Shortest Paths and Vehicle Routing

Petersen, Bjørn; Pisinger, David

Publication date:
2011

Document Version
Publisher's PDF, also known as Version of record

Shortest Paths and Vehicle Routing

PhD thesis 9.2011

DTU Management Engineering

Bjørn Petersen
June 2011

DTU Management Engineering
Department of Management Engineering

Shortest Paths and Vehicle Routing

Ph.D. Thesis

Bjørn Petersen
bjorn@diku.dk

DTU Management Engineering, Technical University of Denmark
Produktionstorvet Bygning 424, DK-2800 Kgs. Lyngby, Denmark

October, 2010

Preface

This Ph.D. thesis was written at the Department of Computer Science, University of Copenhagen (DIKU) from November 2007 to February 2009 and at the Department of Management Engineering, Technical University of Denmark (DTU) from March 2009 to October 2010 under the supervision of Professor David Pisinger.

I would like to thank my colleagues at DIKU and DTU, especially Mads Jepsen and Simon Spoorendonk for the many good discussions and my supervisor David Pisinger for making this possible.

Bjørn Petersen
October 2010

Contents

Part I

Introduction

Chapter 1

Introduction

Bjørn Petersen
DTU Management Engineering, Technical University of Denmark

1 Motivation

Operation Research (OR) is an interdisciplinary branch of applied mathematics and formal science that uses advanced analytical methods such as mathematical modeling, statistical analysis, and mathematical optimization to arrive at optimal or near-optimal solutions to complex decision making problems. OR is often concerned with determining the maximum (of profit, performance, or yield) or minimum (of loss, risk, or cost) of some real-world objective, and strives to support the decision making by providing a number of tools such as mathematical modeling and mathematical programming. Mathematical modeling is used to formulate problems in a concrete way using mathematical equations, whereas mathematical programming covers solution methods for the mathematical formulations. Even though this approach by its nature often has an exponential running time, it has still been very successful, partially due to increased computing power but primarily due to algorithmic improvements. Optimization refers to choosing the best element from a set of available alternatives.

OR is used in many different disciplines including transportation (vehicles, trains, airplanes, ships), production, telecommunication, and finance. In this thesis the focus is on transportation and more concretely on vehicle routing and shortest path problems. Vehicle routing problems, specifically the Capacitated Vehicle Routing Problem (CVRP) and the Vehicle Routing Problem with Time Windows (VRPTW), are interesting because they contain the structure of what makes this type of problems hard to solve. This has historically made CVRP and VRPTW the test ground for new techniques and developments for many other problems.

Through a focus on shortest paths this thesis will mainly look at how to solve vehicle routing problems when they get hard to solve. The difficulty of these problems grows when the solution space grows, e.g., VRPTW instances with loose time windows and large capacities where the possibility of long routes (measured in number of customers) exists. Some instances with these characteristics with as little as 100 customers cannot be solved at present.

1.1 Mathematical Modeling and Programming

Mathematical modeling is defined by a set of variables, used to represent decisions in a problem, and a set of equations (or inequalities) denoted constraints, used to limit the amount of valid decisions. The constraints define the feasible solution space of a problem, i.e., a polytope in a multi-dimensional space that contains all valid solutions to the problem. The objective is a function of the variables that points to the solution(s) in the feasible solution space where the objective function reaches the global optimum.

When the variables are continuous and the constraints and the objective function are linear, the problem is called a linear program (LP). If integrality is imposed on the variables, it is denoted an integer program (IP), and if both types of variables exist, a mixed integer program (MIP) is obtained. In this thesis, problems of the two latter kinds are sought solved, and in that process these problems are relaxed into LP problems.

Many problems can be formulated as (M)IPs and various solution methods have received a lot of attention during the years. The different solution methods can roughly be divided into three categories:

- *Exact algorithms* find solutions that are proven optimal, i.e., no other solution exists with a better objective function value.
- *Heuristics* give no guaranty for the quality of the solution value. They can be useful in cases where running time is an issue and it is not imperative that an optimal solution is found.
- *Approximation algorithms* have bounds on how much their solution can differ from the optimal solution.

Due to the $\mathcal{P} \stackrel{?}{=} \mathcal{NP}$ issues exact solution methods mostly have exponential running time. Nonetheless, the study of exact methods often give insight into the problem behavior that may otherwise be hard to obtain. Furthermore, the improvements of exact methods have pushed the boundaries for what can be solved in reasonable time. Even though heuristic solution methods can speed up exact solution methods the focus in this thesis is solely on exact algorithms.

1.2 Exact Methods

Many exact methods are based on the Branch-and-Bound paradigm, where a relaxation of the problem is used in each node of the branch tree, i.e., an enlarged solution space is considered. If the gap between the lower (LB) and upper bounds (UB) for some node is non-positive it is possible to fathom the sub-tree rooted in that node. Assuming a minimizing objective function, one way of calculating a lower bound is by solving the LP relaxation of the (M)IP defining the problem, i.e., the enlarged solution space allows for integer variables taking on continuous values.

Raising the LB or lowering the UB will make the gap smaller. Still assuming a minimizing objective function, a well studied way to raise the lower bound is by the use of cutting planes. Cutting planes are inequalities that cut off some of the current fractional solution, i.e., the non-integer solution obtained by the LP relaxation. For the raised LB to be a valid LB the inequalities may not cut off any feasible solutions. When incorporating cutting planes into the Branch-and-Bound paradigm a Branch-and-Cut algorithm is obtained.

Some problem formulations may have special structure, i.e., there are variable sets where some constraints are non-overlapping, or a sub-set of constraints is in itself a problem with an effective solution algorithm. In these cases it is possible to apply Dantzig-Wolfe decomposition to divide the problem into smaller subproblems that have their solutions combined in a master problem, see Dantzig and Wolfe [2]. This approach is known as column generation. If the subproblems are solved iteratively (until the master problem objective value cannot improve further) it is called delayed column generation. When incorporating decomposition into the Branch-and-Bound paradigm a Branch-and-Price algorithm is obtained. The subproblems are often problem specific, e.g., shortest path problems which is the focus of this thesis.

It is possible to combine cutting planes with column generation. This is denoted Branch-Cut-and-Price. However, adding cuts is not as straight forward as in the Branch-and-Cut algorithms. The cuts can be divided into two categories:

- Cuts expressed in the original formulation.
- Cuts expressed in the master problem formulation.

The first alternative can be thought of as having the cuts part of the model before decomposition and thereby handling them as any other constraint in the model would be handled. This will in most cases mean that there are some changes in the costs associated with the subproblem but no structural changes, i.e., the same special purpose algorithm can be used without changes to solve the subproblems.

The second alternative may have complicating repercussions for the subproblems, since cuts on master variables do not necessarily map back to the original model, and thus the special structure of the subproblems. The altered subproblems may contain non-linear objective functions and it may be necessary to add additional variables. This may change the complexity of the subproblems and can result in much higher computational efforts being needed. This case is less studied both theoretically and experimentally and is in the context of shortest paths the main focus of this thesis.

2 Goals

The focus of this thesis is on shortest path problems and how to solve them in the context of column and cut generation algorithms, i.e., with negative weights and extra complicating issues to handle costs not directly mappable to the edge weights. The main goals can be summarized as:

- Investigate how to solve shortest path problems in the presence of negative cycles and resource constraints.
- In a column generation context to investigate how to handle effects of cutting planes derived from the master problem formulation.
- Investigate the impact of the cutting planes on the subproblems complexity, on the quality of the lower bounds for the master problem, and the overall running time of the Branch-Cut-and-Price algorithm.
- Explore alternative reformulations to target *difficult* part of problems.

Shortest path problems are present many places, both on its own and as subproblems. According to Dror [3] solving the elementary shortest path problem on a graph containing negative cost cycles is strongly $\mathcal{NP}$-hard.

Many problems decompose into a set partitioning master problem and some kind of shortest path problem. Cuts valid for the problem before the decomposition are often directly applicable to the decomposed model. This is in contradiction to cuts valid for the set partitioning problem which often require some extra handling in the shortest path subproblem. Very efficient cuts are known for the set partitioning polytope including some of the general purpose cut family known as Chvátal-Gomory cuts.

Cuts valid for the set partition polytope are incorporated into the existing subproblem algorithms by modifying these special purpose algorithms. The increased complexity (and thereby potentially increased running times) of the subproblems is a trade-off with the quality of the lower bound obtained in the master problem. The running time saved by exploring fewer branch nodes due to the improved lower bound is hopefully overshadowing the increased effort put in solving the subproblems.

Solving the shortest path problems is often the bottleneck of decomposition algorithms, especially for hard instances. Alternative decompositions target this behavior by moving some of the complexity from the pricing stage to the master problem.

3 Contribution

The main contributions of the thesis, summarized in the points below, is to show

- how to find resource constrained shortest paths by the use of a Branch-and-Cut algorithm.
- how alternative reformulations can be obtained through the use of Partial Paths, so that movement of complexity between master and pricing problem is facilitated.
- theoretically and experimentally how to apply the Chvátal-Gomory cuts of rank 1 known from Branch-and-Cut algorithms for general MIPs to the vehicle routing problem with time windows. Furthermore, to show how to incorporate this into a dynamic programming algorithm for the subproblem. The approach appears very successful and it is possible to solve several previously unsolved instances from the benchmarks of Solomon [7].

A more detailed description of the contributions of each chapter can be found in the following reading guide in Section 4.

4 Reading Guide

This thesis is divided into four parts. The first part consists of this introductory chapter and a chapter on solving Resource Contrained Shortest Paths Problems by Labeling Algorithm. The second part is the main contribution consisting of the most relevant papers produced. The third part sums up the thesis. Finally, the fourth part acts as an appendix and presents contributions that are not within the primary scope of the thesis, but have been performed during the Ph.D. course.

In the following is a chapter-wise guide for reading this thesis.

Chapter 2: Resource Contrained Shortest Paths Problems Solved by a Labeling Algorithm. The chapter presents a general labeling algorithm for solving various resource constrained shortest path problems. A parallelized version of the algorithm is introduced and some brief computational results are presented. When labeling algorithms are applied throughout this thesis this is the algorithm used.

4.1 Part II: Shortest Paths and Vehicle Routing

This part concerns the main topic of the thesis.

Chapter 3: Subset-Row Inequalities Applied to the Vehicle Routing Problem with Time Windows. The paper presents how a subset of the Chvátal-Gomory cuts may be applied to the master problem of a decomposition of the vehicle routing problem with time windows. It is shown how each cut in the master problem increases the complexity of the subproblem and how this is handled in a dynamic programming algorithm. Experimental results were carried out on the Solomon instances and it was possible so solve several previously unsolved instances by this new approach. Furthermore, experiments showed that the cuts improved the lower bounds to an extent that significantly reduced the size of the branch tree. The paper is co-authored with Mads Jepsen, Simon Spoorendonk, and David Pisinger and has been published in the journal Operation Research, see Jepsen et al. [5].

Chapter 4: Chvátal-Gomory Rank 1 Cuts used in a Dantzig-Wolfe Decomposition of the Vehicle Routing Problem with Time Windows. This paper is an extension of the work described in Jepsen et al. [5], and shows how any Chátal-Gomory rank 1 cut can be applied to the vehicle routing problem with time windows. Experimental results show that it was possible to solve even more instances without branching. However, the cut separation times were substantial. The work is co-authored with David Pisinger and Simon Spoorendonk and has been published as a chapter in a book on recent advances within vehicle routing problems, see the chapter by Petersen et al. [6] in the book edited by Golden et al. [4].

Chapter 5: Branch-and-Cut Algorithm for the Elementary Shortest Path Problem with a Capacity Constraint. Elementary shortest path problems with resource constraints occur as a subproblem in many decompositions. This paper presents a very efficient Branch-and-Cut algorithm that regards a single capacity constraint. This is joint work with Mads Jepsen and Simon Spoorendonk. The paper has been submitted for publication.

Chapter 6: Partial Path Column Generation for the Vehicle Routing Problem. This presents a column generation algorithm for the Capacitated Vehicle Routing Problem (CVRP) and the Vehicle Routing Problem with Time Windows (VRPTW). This is joint work with Mads Jepsen and David Pisinger. The paper has been submitted for publication.

Chapter 7: Optimal Routing with Failure Independent Path Protection. This paper presents a practical application of finding shortest paths in the telecommunication industry. The problem consists of finding a collection of paths in a telecommunication network that covers a given bandwidth demand and follows a certain backup policy. Experimental results show

that the implemented backup strategy gives significant bandwidth savings. The paper is co-authored with Thomas K. Stidsen, Simon Spoorendonk, Martin Zachariasen, and Kasper B. Rasmussen and has been published in the journal Networks, see Stidsen et al. [8].

4.2 Part III: Conclusion

This part of the thesis concludes and summarizes on the work presented in Part II.

Chapter 8: Conclusion. This chapter contains the concluding remarks and discussion of potential directions for future research.

Chapter 9: Summary in Danish. This chapter contains a Danish summary of the thesis.

4.3 Part IV: Other Contributions

This part of the thesis presents contributions that are not within the primary scope of the thesis, but have been performed during the Ph.D. course.

Chapter 10: The Simultaneous Vehicle Scheduling and Passenger Service Problem. Passengers using public transport systems often experience waiting times when transferring between two scheduled services. This paper propose a planning approach which seeks to obtain a favorable trade-off between the two contrasting objectives; passenger service and operating cost, by modifying the timetable. The planning approach is referred to as the Simultaneous Vehicle Scheduling and Passenger Service Problem (SVSPSP). The paper is co-authored with Hanne L. Petersen, Allan Larsen, Oli. B. G. Madsen, and Stefan Røpke, and has been submitted for publication.

Chapter 11: The Multi-Commodity k-splittable Maximum Flow Problem. The Multi-Commodity k-splittable Maximum Flow Problem consists of routing as much flow as possible through a capacitated network so that each commodity uses at most k paths and the capacities are satisfied. The problem is solved to optimality through Branch-and-Price. This is joint work with Mette Gamst. The paper has been submitted for publication.

Chapter 12: Partial Path Column Generation for the Elementary Shortest Path Problem with Resource Constraints. As just noted previously, elementary shortest path problems with resource constraints occur as a subproblem in many decompositions. This paper introduces a decomposition of the Elementary Shortest Path Problem with Resource Constraints (ESPPRC), where the path is combined by smaller sub-paths. Computational results by comparing different approaches for the decomposition and comparing the best of these with existing algorithms are shown. It is also shown that the algorithm for many instances outperforms a bidirectional labeling algorithm. This is joint work with Mads Jepsen. The paper has been published as an extended abstract at INOC 2009.

Chapter 13: Partial Path Column Generation for the Vehicle Routing Problem with Time Windows. This paper is related to the work described in Chapter 6 and presents a column generation algorithm for the Vehicle Routing Problem with Time Windows (VRPTW). The traditionally elementary route-columns are relaxed into partial paths, i.e., not necessarily

starting and ending in the depot. This way, the length of the partial path can be bounded and a better control of the size of the solution space for the pricing problem can be obtained. This is joint work with Mads Jepsen. The paper has been published as an extended abstract at INOC 2009.

Chapter 14: The Vehicle Routing Problem Solved by Bounding and Enumeration of Partial Paths. This paper is extended work of Chapter 6, and is inspired by work described by Baldacci et al. [1] where columns with potentially negative reduced cost are enumerated after good upper and lower bounds are found. This is joint work with Mads Jepsen. The paper has been published as an extended abstract at Tristan 2010.

Chapter 15: A solution approach to the ROADEF/EURO 2010 challenge based on Benders Decomposition. The French operations research society, Recherche Opérationnelle et d'Aide à la Décision ROADEF, put forth a challenge to schedule and plan energy production in the French energy sector. An approach based on Bender's Decomposition has been developed. The paper is co-authored with Richard Lusby and Laurent F. Muller.

References

[1] R. Baldacci, N. Christofides, and A. Mingozzi. An exact algorithm for the vehicle routing problem based on the set partitioning formulation with additional cuts. *Mathematical Programming*, 115(2):351–385, 2008. doi: 10.1007/s10107-007-0178-5.

[2] G. B. Dantzig and P. Wolfe. Decomposition principle for linear programs. *Operations Research*, 8(1):101–111, 1960. doi: 10.1287/opre.8.1.101.

[3] M. Dror. Note on the complexity of the shortest path models for column generation in VRPTW. *Operations Research*, 42(5):977–979, 1994. doi: 10.1287/opre.42.5.977.

[4] B. Golden, R. Raghavan, and E. Wasil, editors. *The Vehicle Routing Problem: Latest Advances and New Challenges*. Springer, 2008. doi: 10.1007/978-0-387-77778-8.

[5] M. Jepsen, B. Petersen, S. Spoorendonk, and D. Pisinger. Subset-row inequalities applied to the vehicle-routing problem with time windows. *Operations Research*, 56(2):497–511, 2008. doi: 10.1287/opre.1070.0449.

[6] B. Petersen, D. Pisinger, and S. Spoorendonk. Chvátal-Gomory rank-1 cuts used in a Dantzig-Wolfe decomposition of the vehicle routing problem with time windows. In B. Golden, R. Raghavan, and E. Wasil, editors, *The Vehicle Routing Problem: Latest Advances and New Challenges*, pages 397–420. Springer, 2008. doi: 10.1007/978-0-387-77778-8_18.

[7] M. M. Solomon. Algorithms for the vehicle routing and scheduling problem with time window constraints. *Operations Research*, 35(2):234–265, 1987. doi: 10.1287/opre.35.2.254.

[8] Thomas Stidsen, Bjørn Petersen, Simon Spoorendonk, Martin Zachariasen, and Kasper Bonne Rasmussen. Optimal routing with failure-independent path protection. *Netw.*, 55(2):125–137, 2010. ISSN 0028-3045. doi: http://dx.doi.org/10.1002/net.v55:2.

Chapter 2

Resource Contrained Shortest Paths Problems Solved by a Labeling Algorithm

Bjørn Petersen
DTU Management Engineering, Technical University of Denmark

1 Introduction

The Shortest Path Problem with Resource Constraints (SPPRC) can be stated as: Let $G(V, E)$ with nodes V and edges E be a weighted directed graph, and let R be a set of resources. For each edge $e \in E$ and resource $r \in R$ three parameters are given: A lower limit $a_r(e)$ on the accumulation of resource r when traversing edge $e \in E$; An upper limit $b_r(e)$ on the accumulation of resource r when traversing edge $e \in E$; and an amount $c_r(e)$ of resource r consumed by traversing edge $e \in E$. In general $c_r(e)$ can be a function and can also be dependent on other resources, e.g., $c_r(e, r_1, r_2) : r_1, r_2 \in R$, but will for ease of notation be denoted $c_r(e)$ throughout this chapter. The objective is to find a minimum cost path P, i.e., minimize the cost resource $\bar{c}$, from a source node $o \in V$ to a destination node $d \in V$, where the accumulated resources of P satisfy the limits for all resources $r \in R$. Without loss of generality it is assumed that the limits must be satisfied at the start of each edge e, i.e., before $c_r(e)$ has been consumed. It is noted that equivalent upper and lower limits and consumptions on the nodes can be "pushed" onto the edges, e.g., the outgoing edges of a node.

The Shortest Path Problem with Resource Constraints and k-cycle Elimination (k-cyc-SPPRC) can be stated as the SPPRC but with an additional constraint that the path is k-cycle free. In k-cycle free paths, cycles of size k or smaller are not allowed, i.e., paths containing node sequence $(\ldots, v_0, v_1, \ldots, v_{k-1}, v_0, \ldots)$ are forbidden. The Elementary Shortest Path Problem with Resource Constraints (ESPPRC) can be stated as an SPPRC with an additional constraint that path P is cycle free, i.e., no node $v \in V$ is in P more than once. This is essentially the same as a k-cyc-SPPRC with $k = \infty$. Relaxing the ESPPRC so that all nodes do not have to be elementary gives rise to the Partial Elementary Shortest Path Problem with

Resource Constraints (PESPPRC), where only a subset $S \subseteq V$ of the nodes are not allowed to be in the path more than once. Finally, the integration of k-cyc-SPPRC and PESPPRC demands that the nodes in $S \subseteq V$ are not allowed to be in the path more than once at the same time as none of the other nodes $\bar{S} = V \backslash S$ among themselves forms a cycle of size k or smaller. That is, paths containing node sequence $(\ldots, v_0, e_0, v_1, e_1, \ldots, v_{k-1}, e_{k-1}, v_0, \ldots)$ where $v_i \in \bar{S} \wedge e_i \subseteq S \cup \emptyset : 0 \leq i \leq k-1$ are not allowed.

Dror [8] showed that the ESPPRC is strongly $\mathcal{NP}$-hard, hence a relaxation of the ESPPRC was used as the pricing problem in early BCP algorithms. The Shortest Path Problem with Resource Constraints (SPPRC), first named so by Desrochers [6], can be solved in pseudo-polynomial time, e.g., by use of labeling algorithms. Christofides et al. [4] denoted the SPPRC solutions as q-routes when only a single capacity resource is present. To improve lower bounds of the master problem Desrochers et al. [7] used 2-cycle elimination which was later extended by Irnich and Villeneuve [13] to k-cycle elimination (k-cyc-SPPRC), still with pseudo-polynomial running time.

Beasley and Christofides [1] proposed to solve the ESPPRC using Lagrangian relaxation. However, recently labeling algorithms have become the most popular approach to solve the ESPPRC, see e.g., Dumitrescu [9] and Feillet et al. [10]. When solving the ESPPRC with a labeling algorithm a binary resource for each node is added which increases the complexity of the algorithm compared to solving the SPPRC or the k-cyc-SPPRC. Righini and Salani [17] developed a labeling algorithm using the idea of Dijkstra's bi-directional shortest path algorithm that expands both forward from the source node o and backward from the destination node d and connects paths in the middle, thereby potentially reducing the running time of the algorithm. Furthermore, Righini and Salani [16] and Boland et al. [2] proposed to solve ESPPRC by use of a decremental state space algorithm that iteratively solves a SPPRC by applying resources forcing nodes to be visited at most once. Recently Chabrier [3], Danna and Le Pape [5], and Salani [18] successfully solved several previously unsolved instances of the VRPTW from the benchmarks of Solomon [19] using a labeling algorithm for the ESPPRC.

The chapter is outlined as follows: In Section 2 a quick introduction to the concepts of labeling algorithms as well as a description of how they are applied to general shortest paths are given. Section 3 describes how to make the search for shortest paths bidirectional and a proof of correctness is presented. Section 4 introduces a parallel labeling algorithm. In Section 5 brief computational results are shown. Finally, Section 6 contains some concluding remarks.

2 Labeling Algorithm

Several articles covering the basics of solving shortest path problems by use of labeling algorithms already exist, so it is beyond the scope of this chapter to go into these details. However, a short introduction to settle the notation will be given. For a detailed description see e.g. Irnich [12].

The central part of the algorithm is the use of labels which represent partial paths rooted at node o. Each label has associated a set of attributes:

- A node to which it belongs $\bar{v} \in V$
- A pointer to the label of the parent node p

- The accumulated consumption of each resource $r \in R$ (including the cost resource $\bar{c}$)
- An ordered set of last $k-1$ visited nodes $\pi \subseteq \bar{S}$

Thus, a label L with $\bar{v}(L) = v$ represents a partial path from node o to node v and all the accumulated resources along the path. We will use $f(L)$ to refer to attribute f of a label. E.g. $r(L)$ refers to the accumulated consumption of resource r in label L. The parent $p(L)$ of label L is the label L_p that was extended to create L. L_p is recursively used to find the path $P(L)$ that label L represents. $V(P(L))$ (or shorthand $V(L)$) is the multiset of the predecessors and $E(P(L))$ (or shorthand $E(L)$) are the edges on P. The attributes r and π are not strictly necessary and are only present for notational and computational reasons, they can always be computed by following the chain of parent labels $L_p, L_{p-1}, ..., L_o$ back to the starting node o.

In the following it is assumed that all resources are bounded strongly from above, and weakly from below, i.e., if the current resource accumulation is below the lower limit on a given edge e, it is allowed to fill up the resource to the lower limit, e.g., waiting for a time window to open. This means that two consecutive labels L_u and L_v related by an edge $e = (u, v)$, i.e., L_u is extended and creates L_v, where $\bar{v}(L_u) = u$ and $\bar{v}(L_v) = v$, must satisfy

$$r(L_v) \leq b_r(e) \qquad \forall r \in R \tag{1}$$

$$r(L_v) = \max\{r(L_u) + c_r(e), a_r(e)\} \qquad \forall r \in R \tag{2}$$

$$v \neq w \qquad \forall w \in \pi(L_u) \tag{3}$$

Here (1) demands that L_v satisfies the upper limit of resource r corresponding to edge $e = (u, v)$, while (2) states that resource r at label L_v corresponds to the resource consumption at label L_u plus the amount consumed by traversing edge e, respecting the lower limit on edge e and (3) ensures no cycles of size smaller than k.

The concept of labeling algorithms is to iteratively extend labels (according to (1)–(3)) in the following way, until there are no more labels left. When a label has been extended it, is considered treated:

```
LABELING(G, o, d)
 1  L_init = FIRST-LABEL(o)
 2  PQ.ENQUEUE(L_init)
 3  while PQ ≠ ∅
 4      REMOVE-DOMINATED(PQ)
 5      L = PQ.DEQUEUE()
 6      for each node v ∈ EXTENDABLES(L)    // Nodes to which L can be extended
 7          L_v = EXTEND-LABEL(L, v)
 8          if v̄(L_v) = d
 9              STORE-SOLUTION(L_v, sol)
10          else PQ.ENQUEUE(L_v)
11  return sol
```

Line 1 makes the first label which is then put in a queue in line 2. Lines 4–10 loop as long as there are untreated labels left. A label is selected in line 5 which is then extended in line 7. If the new label represents a path from o to d, it is stored in line 9. Otherwise it is put in the queue for later treatment.

From the pseudocode it is clear that without REMOVE-DOMINATED in line 4 this results in a complete enumeration of all feasible paths.

2.1 Dominance

The goal of dominance is to reduce the number of labels that are created during the execution of the labeling algorithm, since it is not desirable to extend labels that are not part of an optimal solution. Unfortunately, it is not known in advance which labels span optimal solutions, but it might be possible to decide for some labels that they are not part of any optimal solution. If just any optimal solution is sought, dominance is to reduce the number of labels extended and still be able to find an optimal solution. A label is thus said to be dominated if its removal during the run of the algorithm does not remove all optimal solution.

In the following it is assumed that all the extension functions $c_r(e)$ are non-decreasing. A non-decreasing function $c_r(e)$ has the following property:

Definition 1. A function f is non-decreasing iff:

$$x \leq y \Rightarrow f(x) \leq f(y) \quad \forall x, y$$

In relation to dominance it is necessary to consider extensions of labels. For this reason three definitions are presented (slightly modified from Irnich and Villeneuve [13]):

Definition 2. The set of all feasible paths from label L to node u considering the resource consumption of label L is defined as $\mathcal{F}(L, u)$.

Definition 3. The set of all feasible paths from label L to node u considering the k-cycle elimination and the partial elementarity is defined to be $\mathcal{S}(L, u)$.

Definition 4. All feasible extensions of label L is defined as:

$$\mathcal{E}(L) = \mathcal{F}(L, t) \cap \mathcal{S}(L, t)$$

With Definition 4 as a building block the following definition of domination is now given:

Definition 5. A set of labels $\mathcal{L}_i$ dominates label L_j if:

$$\bar{v}(L_i) = \bar{v}(L_j) \qquad \forall L_i \in \mathcal{L}_i \tag{4}$$

$$\bar{c}(L_i) \leq \bar{c}(L_j) \qquad \forall L_i \in \mathcal{L}_i \tag{5}$$

$$\mathcal{E}(L_j) \subseteq \bigcup_{L_i \in \mathcal{L}_i} \mathcal{E}(L_i) \tag{6}$$

In other words, the paths corresponding to labels in $\mathcal{L}_i$ and the path L_j should end at the same node $\bar{v}(L_i) = \bar{v}(L_j) \in V : \forall L_i \in \mathcal{L}_i$, each path corresponding to some label $L_i \in \mathcal{L}_i$ should cost no more than the path corresponding to label L_j, and finally any feasible extension of L_j is also a feasible extension of some $L_i \in \mathcal{L}_i$.

Definition 5 implies that if L_j is dominated then any path $P(L_j, \epsilon)$ consisting of L_j concatenated with a feasible extension $\epsilon \in \mathcal{E}(L_j)$ is not a unique optimal solution, since at least one other label $L_i \in \mathcal{L}_i$ can also be concatenated with ϵ and make a path $P(L_i, \epsilon)$ that is as least as cheap, because $\bar{c}(L_i) + c_{\bar{c}}(\epsilon, r(L_i)) \leq \bar{c}(L_j) + c_{\bar{c}}(\epsilon, r(L_j))$ due to Definition 1, that is:

$$\begin{aligned} & \forall \epsilon \in \mathcal{E}(L_j) \quad \exists L_i \in \mathcal{L}_i \ : \ \epsilon \in \mathcal{E}(L_i) \ \wedge \ \bar{c}(L_i) \leq \bar{c}(L_j) \\ \Rightarrow\ & \forall \epsilon \in \mathcal{E}(L_j) \quad \exists L_i \in \mathcal{L}_i \ : \ \epsilon \in \mathcal{E}(L_i) \ \wedge \ \bar{c}(P(L_i, \epsilon)) \leq \bar{c}(P(L_j, \epsilon)) \end{aligned}$$

Each node in the set of elementary nodes S, i.e., the nodes that can only be visited once, can be modeled using a binary resource. Feillet et al. [10] suggested to consider the set of nodes in S that cannot be reached from a label L_i and compare the set with the unreachable nodes of a label L_j in order to determine if some extensions are impossible. Or in other words: update the node resources in an eager fashion instead of a lazy. The following definition is a generalization of Feillet et al. [10][Definition 3].

Definition 6. Given a start node $o \in V$ and a label L with $\bar{v}(L) = u$, a node $v \in V$ is considered *unreachable* if v has already been visited on the path from o to u, i.e., $v \in V(L)$ or if a resource window is violated, e.g.:

$$\exists r \in R \qquad\qquad r(L) + \ell_r(u, v) > b_r(v)$$

where $\ell_r(u, v)$ is a lower bound on the consumption of resource r on all feasible paths from u to v. The *node resources* are then given as: $v(L) = 1$ indicates that node $v \in V$ is unreachable from node $\bar{v}(L) \in V$, and $v(L) = 0$ otherwise.

In the following $\bar{\mathcal{E}}(\pi(L))$ (or shorthand $\bar{\mathcal{E}}(L)$) will be the set of all feasible extensions for label L only considering the k-cycle elimination constraint, and is equivalent to the concept of *Hole-Sets* as defined by Irnich [12].

To determine if (6) holds can be quite cumbersome, as the straightforward definition suggests that we calculate all extensions of the involved labels. Therefore a sufficient criteria for (6) is sought which can be computed faster. If label L_i has consumed less resources than label L_j, then no resources are limiting the possibilities of extending L_i compared to L_j, hence the following proposition can be used as a restricted version of the dominance criteria in Definition 5.

Proposition 1 (Sufficient condition)**.** A set of labels $\mathcal{L}_i$ dominates label L_j if:

$$\bar{v}(L_i) = \bar{v}(L_j) \qquad \forall L_i \in \mathcal{L}_i \tag{7}$$

$$r(L_i) \leq r(L_j) \qquad \forall r \in R, \forall L_i \in \mathcal{L}_i \tag{8}$$

$$\bar{\mathcal{E}}(L_j) \subseteq \bigcup_{L_i \in \mathcal{L}_i} \bar{\mathcal{E}}(L_i) \tag{9}$$

and node resources are set according to Definition 6.

Proof. We check Definition 5. Equation (4) follows directly from (7) and (5) follows from (8) with $r = \bar{c}$, i.e., the cost resource. The remaining concern is if (6) holds for $\mathcal{L}_i$ and L_j. The proof is by contradiction. Assume that (7), (8), and (9) are satisfied but that (6) is not satisfied. Then an extension $\epsilon \in \mathcal{E}(L_j) \backslash \bigcup_{L_i \in \mathcal{L}_i} \mathcal{E}(L_i)$ must exist which is feasible for L_j but not for any $L_i \in \mathcal{L}_i$. Let L^u denote the label that is obtained with $\bar{v}(L^u) = v_u$ after L_j has recursively been extended through ϵ, let $\mathcal{L}^u$ be equivalently defined, let $v_1, \ldots, v_{h-1}, v_h, \ldots$ be the nodes on ϵ, and let v_h be the first node on ϵ preventing the extension of all $L_i^{h-1} \in \mathcal{L}_i^{h-1}$. There are only three conditions where this can happen for each $L_i^{h-1} \in \mathcal{L}_i^{h-1}$:

1) $v_h(L_i^{h-1}) = 1$

2) $\exists r \in R, \qquad r(L_i^{h-1}) + l_r(v_{h-1}, v_h) > b_r(h)$

3) $\epsilon \notin \bar{\mathcal{E}}(L_i)$

Since L_j can be extended with ϵ, the equivalent conditions for L_j^{h-1} are:

1) $v_h(L_j^{h-1}) = 0$

2) $r(L_j^{h-1}) + c_r(v_{h-1}, v_h) \leq b_r(h), \qquad \forall r \in R$

3) $\epsilon \in \bar{\mathcal{E}}(L_j)$

Since all resources are consumed according to Definition 1 on ϵ until v_{h-1} for all $L_i \in \mathcal{L}_i$ and L_j, the above conditions contradict that (7) and (8) are satisfied. Moreover, $\epsilon \in \bar{\mathcal{E}}(L_j)$ and $\epsilon \notin \bar{\mathcal{E}}(L_i)$ contradict that (9) is satisfied. Hence, $\mathcal{E}(L_j) \setminus \bigcup_{L_i \in \mathcal{L}_i} \mathcal{E}(L_i) = \emptyset$, which implies $\mathcal{E}(L_j) \subseteq \bigcup_{L_i \in \mathcal{L}_i} \mathcal{E}(L_i)$, and (6) holds. That is, Definition 5 holds and $\mathcal{L}_i$ dominates L_j. □

Using Proposition 1 as a dominance criteria is a restriction of the dominance criteria of Definition 5 since only a subset of labels satisfying (7), (8), and (9) satisfies (4), (5), and (6). It is noted that Condition (8) can be tightened by being lazy with $r(L_i)$ and eager with $r(L_j)$. Furthermore, if $k \leq 1$ Condition (9) is automatically satisfied so $|\mathcal{L}_i| = 1$.

Decreasing extension-functions can always be handled by use of equality on the affected resources, but can be tightened if a lower and an upper bound in known, see Reinhardt and Pisinger [15] for further details. Being more aggressive in REMOVE-DOMINATED, i.e., removing non-dominated labels, yields a heuristic solution but with likely improved running time.

3 Bidirectional Search

The concept of bidirectionality is to look for the shortest path from node o to node d by finding paths from o to 'the middle' and 'reverse paths' from d to 'the middle'. The paths meeting in 'the middle' are then spliced together, and thereby a shortest path is obtained. 'The middle' is defined by the consumption of a monotone resource r_{mono}, i.e., $c_{r_{mono}}$ is either non-negative or non-positive. Furthermore, it is required that all cycles defined by $c_{r_{mono}}$ are non-zero.

The reason for doing this for ESPPRC is to halve the exponential factor in the worst case number of labels, e.g., $O(V!2^V)$ can be reduced to $O(\frac{V}{2}!2^{V/2})$ by selecting r_{mono} as the number of visited nodes. For k-cyc-SPPRC and pure SPPRC the theoretical worst case number of labels is not affected but a better practical running time is hoped for. For PESPPRC the worst case number of labels is dependent on the number of nodes in S and will be somewhere in between that of SPPRC and ESPPRC.

The bidirectional algorithm consists of the following three parts:

- Find (part of) the shortest *forward* path going from o towards d at the same time as finding (part of) the shortest *backward* 'reverse path' going from d towards o.
- Combine a forward label L_f and backward label L_b with $\bar{v}(L_f) = \bar{v}(L_b)$ to obtain a path $P(L_f, L_b)$.
- Stop at the 'middle', e.g., stop when the consumption of resource r_{mono} in a label reaches x_{stop}, where $\min_{v \in V}(a_{r_{mono}}(v)) \leq x_{stop} \leq \max_{v \in V}(b_{r_{mono}}(v))$.

Forward and Backward Paths

The algorithm from Section 2 can be reversed by starting with a label L_b in node d with the consumption of each resource set to the upper bound $r(L) = b_r(d)$ for all $r \in R$. Then go towards node o and treat extensions and dominance equivalently – this of course is only possible if an inverse of the extension function exists. The algorithm from Section 2 will be referred to as the *forward algorithm* and the reversed counterpart will be referred to as the *backward algorithm*. Using equivalent argumentation as for the forward algorithm it is clear that the backward algorithm also yields optimal solutions to the problems.

The bidirectional algorithm works by running a forward algorithm together with a backward algorithm keeping two sets of labels: The forward labels $\mathcal{L}^f$ and the backward labels $\mathcal{L}^b$. The following pseudocode shows how the bidirectional algorithm works.

```
BiDirectional-Labeling(G, o, d)
 1  L_o = First-Label-Forward(o)
 2  L_d = First-Label-Backward(d)
 3  PQ_f.enqueue(L_o)
 4  PQ_b.enqueue(L_d)
 5  while PQ_f ≠ ∅  or PQ_b ≠ ∅
 6      if PQ_f.size() < PQ_b.size()
 7          Remove-Dominated(PQ_f)
 8          L = PQ_f.dequeue()
 9      else Remove-Dominated(PQ_b)
10          L = PQ_b.dequeue()
11      for each node v ∈ Extendables(L)
12          L_v = Extend-Label(L, v)
13          PQ.enqueue(L_v)
14          for each label L ∈ Spliceable(L_v)
15              path = Splice(L_v, L)
16              Store-Solution(path, sol)
17  return sol
```

As before, after the initial labels are created and enqueued the algorithm loops until no labels are left untreated. The functions in the pseudocode have knowledge about the direction (forward or backward) and behave accordingly. Line 6 lets the two directions grow in parallel.

Disregarding the stopping criterion, it is clear that the algorithm will find at least two optimal paths. One is found going forward and one is found going backwards. These two paths may be identical.

Splicing the Paths

At any time during the execution of the algorithm above there are two sets of labels $\mathcal{L}_v^f : L \in \mathcal{L}^f \ \wedge \ \bar{v}(L) = v$ and $\mathcal{L}_v^b : L \in \mathcal{L}^b \ \wedge \ \bar{v}(L) = v$ belonging to each node $v \in V$. Consider a label $L_f \in \mathcal{L}_v^f$ and a label $L_b \in \mathcal{L}_v^b$. If the sub-path L_b is in the extension $L_b \in \mathcal{E}(L_f)$ of L_f, the two labels can be combined to form a feasible solution P, this is denoted *splicing*. Since a path may use several nodes, a given path P may be the product of several different splicings, e.g., one for each of the $|P|$ nodes in P.

For obvious reasons it is desirable only to get unique paths, so when searching for a path P, two labels $L_f \in \mathcal{L}_v^f$ and $L_b \in \mathcal{L}_v^b$ in P with $\bar{v}(L_f) = \bar{v}(L_b)$ are only spliced when $\bar{v}(L_f) = v$ is a unique node $v \in V(P)$ on P. One way to find this unique node v to splice at was proposed by Righini and Salani [17] and is defined as the node $v \in V(P)$ where L_f and L_b are as close as possible to having the same consumption of r_{mono}. A tie is broken arbitrarily, e.g., L_f takes priority.

Consequently, we propose another way to find the unique node v. If more than half the upper limit

$$x_{stop} = \left\lceil \frac{\max_{v \in V}(b_{r_{mono}}(v))}{2} \right\rceil$$

of resource r_{mono} is consumed on path P, one edge $(i, j) \in E(P)$ either crosses x_{stop} or ends at x_{stop}, choosing node j as the splicing point for P will be unique. If the consumption of $r_{mono}(P) \leq x_{stop}$, choosing the first (or the last) node of P as splicing point will be unique.

Stop at the Middle

It is clear that if $r_{mono}(P) > x_{stop}$ then at least one sub-path from the forward algorithm or one sub-path from the backward algorithm has to contain the edge $(i, j) \in E(P)$ that crosses the 'middle'. Furthermore, at least one of them has to contain the first (or last) edge.

From the description of splicing nodes above, it is clear that there is no reason, for the algorithm without the responsibility of crossing, to extend a label if a consumption of more than x_{stop} will be obtained. For the algorithm with the responsibility of crossing, there is no reason to extend a label further when a consumption of x_{stop} is obtained. Therefore, both algorithms can be stopped early and an optimal path P is still found.

Proposition 2. The bidirectional algorithm returns an optimal solution for any value of x_{stop}.

Proof. Without loss of generality assume that the forward algorithm crosses x_{stop} if $r_{mono}(P) > x_{stop}$, the last node is chosen for splicing if $r_{mono}(P) \leq x_{stop}$, and the optimal path P is unique. Let $P = v_1 \to \ldots \to v_n$, let $L_f^i : \bar{v}(L_f^i) = v_i,\ \forall v_i \in V(P)$ be the labels representing P for the forward algorithm, and let $L_b^i : \bar{v}(L_b^i) = v_i,\ \forall v_i \in V(P)$ be the labels representing P for the backward algorithm.

The proof is by contradiction. Assume that the optimal path P is not found. This can only happen in three cases:

1) For some node $v_i \in V(P)$ neither L_f^i nor L_b^i is created.

2) For some node $v_i \in V(P)$ neither L_f^i nor L_b^i exist after domination.

3) There is no node $v_i \in V(P)$ where both L_f^i and L_b^i exist after domination.

It will now be shown that none of the three cases can happen.

Since both the forward and the backward algorithm find P, for 'Case 1' to happen the stopping criteria must have stopped both of them before node v_i was reached. This means that $r_{mono}(L_f^i) > x_{stop}$ and $r_{mono}(L_b^i) \leq x_{stop}$ thus $L_f^i \notin E(L_b^i)$ which contradicts that P is feasible.

For 'Case 2' to happen at least one of L_f^i and L_b^i must have been deleted during domination, which is in contradiction with Definition 5 or that P is unique and optimal.

'Case 3' can be divided into two cases: one where $r_{mono}(P) \leq x_{stop}$ and another where $r_{mono}(P) > x_{stop}$. If $r_{mono}(P) \leq x_{stop}$, then the splicing must be done at v_n. L_b^n clearly exists, so L_f^n must be absent. This can only happen when $r_{mono}(L_f^{n-1}) > x_{stop}$ which contradicts that $r_{mono}(P) \leq x_{stop}$. If $r_{mono}(P) > x_{stop}$, then there must be a node $v_i \in V(P)$ where L_b^i cannot be extended more due to $r_{mono}(L_b^i) - r_{mono}(e(v_{i-1}, v_i)) = r_{mono}(L_b^{i-1}) \leq x_{stop}$. Since L_f^i does not exist, $r_{mono}(L_f^{i-1}) > x_{stop}$. This means that $r_{mono}(L_f^{i-1}) > x_{stop} \geq r_{mono}(L_b^{i-1})$ implying $L_f^{i-1} \notin E(L_b^{i-1})$ which contradicts that P is feasible. □

Since any value of x_{stop} yields an optimal solution, x_{stop} can be adjusted to balance the amount of labels created by the forward and the backward algorithms respectively. As long as $x_{stop} \leq \min(PQ_b)$ the value of x_{stop} can be raised.

4 Parallel Labeling Algorithm

The algorithm just described is a so-called *pushing* algorithm because labels are extended *from* a node to neighbouring nodes. A slightly different variant is a *pulling* algorithm where labels are extended *to* a node from neighbouring nodes. Pulling nodes have a slightly different structure that facilitates parallelization. Going from a label pushing to a label pulling approach only takes a little rearranging of the pseudo-code and a priority queue for each node.

PARALLEL-LABELING(G, o, d)

```
 1  PQ_o.ENQUEUE(FIRST-LABEL(o))
 2  while ∃v ∈ V : PQ_v ≠ ∅
 3      for each node i ∈ V
 4          for each node j ∈ V
 5              L_i = PQ_j.GETSOME()
 6          for each L ∈ L_i : EXTENDABLE(L, i)
 7              L_i = EXTEND-LABEL(L, i)
 8              if v̄(L_i) = d
 9                  STORE-SOLUTION(L_i, sol)
10              else PQ_i^temp.ENQUEUE(L_i)
11          REMOVE-DOMINATED(PQ_i^temp)
12      for each node i ∈ V
13          PQ_i.DELETESOME()
14          PQ_i.ADD(PQ_i^temp)
15  return sol
```

As long as there is an untreated label left, each node tries to pull in labels, which have not consumed too much of r_{mono}, from neighbouring nodes. The consumption check is performed in the GETSOME() function in line 5. Domination is performed in line 11 after the new labels are created. When all nodes are finished pulling in labels, the priority queues are updated in lines 13–14.

Since only local data are changed for each node $i \in V$ in the lines 3–11, they can be run in parallel.

5 Computational Results

A bidirectional parallelized label-pulling algorithm has been implemented in C++ with GCC [11] as compiler. POSIX thread [14] is used as means of obtaining concurrency. Binary min_heaps have been used for priority queues.

Only brief computational results are shown here, since the parallel bi-directional labeling algorithm presented in this chapter is used in the following chapters and the performance is documented there. The computational evaluation has been performed on a dual 2.66GHz Intel® Xeon® X5355 machine with 16 GB of RAM. Table 1 shows the running times and speedup for two different kinds of ESPPRC.

Instance	T_1	T_2	Speedup_2	T_4	Speedup_4	T_8	Speedup_8
A-n61-k9	3.05	2.19	1.39	1.80	1.69	1.71	1.78
A-n69-k9	6.48	4.83	1.34	3.81	1.70	3.36	1.93
B-n50-k8	7.61	5.32	1.43	4.47	1.70	4.09	1.86
C203.100	4.96	3.75	1.32	3.44	1.44	3.27	1.52
R112.100	2.84	1.95	1.46	1.66	1.71	1.35	2.10
R203.100	5.51	3.75	1.47	3.14	1.75	2.60	2.12
R204.50	163.06	138.58	1.18	95.29	1.71	75.74	2.15
R206.100	14.13	8.72	1.62	6.71	2.11	5.30	2.67
R210.100	15.16	9.65	1.57	7.68	1.97	6.37	2.38
RC203.100	13.80	9.46	1.46	8.01	1.72	7.38	1.87
RC206.100	1.18	0.89	1.33	0.84	1.40	0.71	1.66
RC207.100	8.34	5.27	1.58	4.09	2.04	3.35	2.49
Average			1.43		1.75		2.04

Table 1: ESPPRC solved by parallel bi-directional labeling algorithm. The A* and B* instances have a single load resource, whereas the C*, R*, and RC* have a load as well as a time resource. T_i is the time in seconds when run on i cores. Speedup_i is the relative speedup from one to i cores.

It can be concluded that some speedup is present. It can also be concluded that more cores give larger speedup. The speedup is not linear in the number of cores, which can be explained by limited memory bus speed. An average speedup of more than two must be considered satisfactory.

6 Concluding Remarks

A general labeling algorithm for solving various resource constrained shortest path problems has been presented. A parallel version was introduced and some computational results were presented that showed that a speedup is experienced when running on multiple cores.

References

[1] J.E. Beasley and N. Christofides. An algorithm for the resource constrained shortest path problem. *Networks*, 19:379–394, 1989. doi: 10.1002/net.3230190402.

[2] N. Boland, J. Dethridge, and I. Dumitrescu. Accelerated label setting algorithms for the elementary resource constrained shortest path problem. *Operation Research Letters*, 34 (1):58–68, 2006. doi: 10.1016/j.orl.2004.11.011.

[3] A. Chabrier. Vehicle routing problem with elementary shortest path based column generation. *Computers & Operations Research*, 33(10):2972–2990, 2006. doi: 10.1016/j.cor.2005.02.029.

[4] N. Christofides, A. Mingozzi, and P. Toth. Exact algorithms for the vehicle routing problem, based on spanning tree and shortest path relaxations. *Mathematical Programming*, 20(1):255–282, Dec 1981. doi: 10.1007/BF01589353.

[5] E. Danna and C. Le Pape. Branch-and-price heuristics: A case study on the vehicle routing problem with time windows. In G. Desaulniers, J. Desrosiers, and M. M. Solomon, editors, *Column Generation*, chapter 4, pages 99–129. Springer, 2005. doi: 10.1007/0-387-25486-2_4.

[6] M. Desrochers. *La fabrication d2019horaires de travail pour les conducteurs d2019autobus par une mÃ©thode de gÃ©nÃ©ration de colonnes.* PhD thesis, UniversitÃ© de MontrÃ©al, MontrÃ©al, Canada, 1986.

[7] M. Desrochers, J. Desrosiers, and M. Solomon. A new optimization algorithm for the vehicle routing problem with time windows. *Operations Research*, 40(2):342–354, 1992. doi: 10.1287/opre.40.2.342.

[8] M. Dror. Note on the complexity of the shortest path models for column generation in VRPTW. *Operations Research*, 42(5):977–979, 1994. doi: 10.1287/opre.42.5.977.

[9] I. Dumitrescu. *Constrained Path and Cycle Problems.* PhD thesis, Department of Mathematics and Statistics, University of Melbourne, Australia, 2002.

[10] D. Feillet, P. Dejax, M. Gendreau, and C. Gueguen. An exact algorithm for the elementary shortest path problem with resource constraints: Application to some vehicle routing problems. *Networks*, 44(3):216–229, 2004. doi: 10.1002/net.v44:3.

[11] GCC. Gnu Compiler Collection version 4.4.3, 2010. http://gcc.gnu.org/.

[12] S. Irnich. Resource extension functions: Properties, inversion, and generalization to segments. *OR Spectrum*, 30(1):113–148, 2008. doi: 10.1007/s00291-007-0083-6.

[13] S. Irnich and D. Villeneuve. The shortest path problem with resource constraints and k-cycle elimination for $k \geq 3$. *INFORMS Journal on Computing*, 18:391–406, 2006. doi: 10.1287/ijoc.1040.0117.

[14] POSIX thread. IEEE Std 1003.1, 2004 Edition, 2004. http://www.unix.org/version3/ieee_std.html.

[15] L. B. Reinhardt and D. Pisinger. Multi-objective and multi-constrained non-additive shortest path problems. *Computers & Operations Research*, 38(3):605 – 616, 2011. ISSN 0305-0548. doi: DOI:10.1016/j.cor.2010.08.003.

[16] G. Righini and M. Salani. New dynamic programming algorithms for the resource constrained shortest path problem. Technical Report 69, Note del Polo - Ricerca, Dipartimento di Tecnologie dell'Informazione, Universitá degli studi di Milano, 2005.

[17] G. Righini and M. Salani. Symmetry helps: bounded bi-directional dynamic programming for the elementary shortest path problem with resource constraints. *Discrete Optimization*, 3(3):255–273, 2006. doi: 10.1016/j.disopt.2006.05.007.

[18] M. Salani. *Branch-and-Price Algorithms for Vehicle Routing Problems*. PhD thesis, Universitá Degli Studi Di Milano, Facoltá di Scienza Matematiche, Fisuche e Naturali Dipartimento di Technologie dell'Informazione, Milano, Italy, 2005.

[19] M. M. Solomon. Algorithms for the vehicle routing and scheduling problem with time window constraints. *Operations Research*, 35(2):234–265, 1987. doi: 10.1287/opre.35.2.254.

Part II

Shortest Paths and Vehicle Routing

Chapter 3

Subset-Row Inequalities Applied to the Vehicle Routing Problem with Time Windows

Mads Jepsen
DIKU Department of Computer Science, University of Copenhagen

Bjørn Petersen
DIKU Department of Computer Science, University of Copenhagen

Simon Spoorendonk
DIKU Department of Computer Science, University of Copenhagen

David Pisinger
DIKU Department of Computer Science, University of Copenhagen

Abstract

This paper presents a branch-and-cut-and-price algorithm for the vehicle routing problem with time windows. The standard Dantzig-Wolfe decomposition of the arc flow formulation leads to a set partitioning problem as the master problem and an elementary shortest path problem with resource constraints as the pricing problem. We introduce the subset-row inequalities, which are Chvatal-Gomory rank-1 cuts based on a subset of the constraints in the master problem. Applying a subset-row inequality in the master problem increases the complexity of the label-setting algorithm used to solve the pricing problem since an additional resource is added for each inequality. We propose a modified dominance criterion that makes it possible to dominate more labels by exploiting the step-like structure of the objective function of the pricing problem. Computational experiments have been performed on the Solomon benchmarks where we were able to close several instances. The results show that applying subset-row inequalities in the master problem significantly improves the lower bound, and in many cases makes it possible to prove optimality in the root node.

Published in the journal *Operations Research*, 2008

1 Introduction

The vehicle routing problem with time windows (VRPTW) can be described as follows: A set of customers, each with a demand, needs to be serviced by a number of vehicles all starting and ending at a central depot. Each customer must be visited exactly once within a given time window, and the capacity of the vehicles must not be exceeded. The objective is to service all customers traveling the least possible distance. In this paper we consider a homogenous fleet, i.e., all vehicles are identical.

The standard Dantzig-Wolfe decomposition of the arc flow formulation of the VRPTW is to split the problem into a master problem (a set partitioning problem) and a pricing problem (an elementary shortest path problem with resource constraints (ESPPRC), where capacity and time are the constrained resources). A restricted master problem can be solved with delayed column generation and embedded in a branch-and-bound framework to ensure integrality. Applying cutting planes either in the master or the pricing problem leads to a branch-and-cut-and-price algorithm (BCP).

Kohl et al. [23] implemented a successful BCP algorithm for the VRPTW by applying subtour elimination constraints and *two-path* cuts. Cook and Rich [8] generalized the two-path cuts to the *k-path* cuts. Common for these BCP algorithms is that all applied cuts are valid inequalities for the VRPTW, i.e., the *original* arc flow formulation, and contain a structure making it possible to handle values of the dual variables in the pricing problem without increasing the complexity of the problem. Fukasawa et al. [17] refer to this as a *robust* approach in their paper, where a range of valid inequalities for the capacitated vehicle routing problem are used in a BCP algorithm. The topic of column generation and BCP algorithms has been surveyed by Barnhart et al. [1] and LÃ$\frac{1}{4}$bbecke and Desrosiers [27].

Dror [13] showed that the ESPPRC is strongly $\mathcal{NP}$-hard, hence a relaxation of the ESPPRC was used as a pricing problem in earlier BCP approaches for the VRPTW. The relaxed pricing problem where non-elementary paths are allowed is denoted the shortest path problem with resource constraints (SPPRC) and can be solved in pseudo-polynomial time using a label-setting algorithm, which was initially done by Desrochers [11]. To improve lower bounds of the master problem, Desrochers et al. [12] used 2-cycle elimination, which was later extended by Irnich and Villeneuve [20] to k-cycle elimination (k-cyc-SPPRC) where cycles containing k or less nodes are not permitted.

Beasley and Christofides [2] proposed to solve the ESPPRC using Lagrangian relaxation. However, recently label-setting algorithms have become the most popular approach to solve the ESPPRC; see e.g. Dumitrescu [14] and Feillet et al. [16]. When solving the ESPPRC with a label-setting algorithm a binary resource for each node is added, which increases the complexity of the algorithm compared to solving the SPPRC or the k-cyc-SPPRC. Righini and Salani [32] developed a label-setting algorithm using the idea of Dijkstra's bi-directional shortest path algorithm that expands both forward and backward from the depot and connects routes in the middle, thereby potentially reducing the running time of the algorithm. Furthermore Righini and Salani [32] and Boland et al. [3] proposed a decremental state space algorithm that iteratively solves a SPPRC by applying resources that force nodes to be visited at most once. Recently Chabrier [5], Danna and Le Pape [9], and Salani [33] successfully solved several previously unsolved instances of the VRPTW from the benchmarks of Solomon [34] using a label-setting algorithm for the ESPPRC.

In this paper, we extend the BCP framework to include valid inequalities for the master problem, more specifically by applying the subset-row (SR) inequalities to the set partitioning

master problem. Nemhauser and Park [28] developed a similar BCP algorithm for the edge coloring problem, but to our knowledge no such algorithms for the VRPTW have been presented. Applying the SR inequalities leads to an increased complexity of the pricing problem since each inequality is represented by an additional resource. To improve the performance of the label-setting algorithm, we introduce a modified dominance criterion that handles the reduced cost calculation in a reasonable way. Moreover, the SR inequalities potentially provide better lower bounds and smaller branch trees.

The paper is organized as follows: In Section 2 we give an overview of the Dantzig-Wolfe decomposition of the VRPTW and describe how to calculate the reduced cost of columns when column generation is used. In Section 3 we introduce the SR inequalities and show that the separation problem is $\mathcal{NP}$-complete. In Section 4 we review the basics of a label-setting algorithm for solving the ESPPRC and show how to handle the modified pricing problem in the same label-setting algorithm. For details regarding label-setting algorithms (including bi-directionality) we refer to Desaulniers et al. [10], Irnich and Desaulniers [19], Irnich [18], Righini and Salani [31]. An algorithmic outline and computational results, using the Solomon benchmark instances, are presented in Section 5. Section 6 concludes the paper.

2 Decomposition

Let C be the set of customers, let the set of nodes be $V = C \cup \{o, o'\}$ where $\{o\}$ denotes the depot at the start of the routes and $\{o'\}$ denotes the depot at the end; and let $E = \{(i,j) : i,j \in V,\ i \neq j\}$ be the edges between the nodes. Let K be the set of vehicles with $|K|$ unbounded, each vehicle having capacity D, and let d_i be the demand of customer $i \in C$ and $d_o = d_{o'} = 0$. Let a_i be the beginning and b_i be the end of the time window for node $i \in V$. Let s_i be the service time for $i \in V$ and let t_{ik} be the time vehicle $k \in K$ visits node $i \in V$, if k visits i. Let c_{ij} be the travel cost on edge $(i,j) \in E$ and let x_{ijk} be a variable indicating whether vehicle $k \in K$ traverses edge $(i,j) \in E$. Last let $\tau_{ij} = c_{ij} + s_i > 0$ be the travel time on edge $(i,j) \in E$ plus the service time of customer i. The three-index flow model (Toth and Vigo [36]) for the VRPTW is:

$$\min \sum_{k \in K} \sum_{(i,j) \in E} c_{ij} x_{ijk} \tag{1}$$

$$\text{s.t.} \sum_{k \in K} \sum_{(i,j) \in \delta^+(i)} x_{ijk} = 1 \qquad \forall i \in C \tag{2}$$

$$\sum_{(i,j) \in \delta^+(o)} x_{ijk} = \sum_{(i,j) \in \delta^-(o')} x_{ijk} = 1 \qquad \forall k \in K \tag{3}$$

$$\sum_{(j,i) \in \delta^-(i)} x_{jik} - \sum_{(i,j) \in \delta^+(i)} x_{ijk} = 0 \qquad \forall i \in C,\ \forall k \in K \tag{4}$$

$$\sum_{(i,j) \in E} d_i x_{ijk} \leq D \qquad k \in K \tag{5}$$

$$a_i \leq t_{ik} \leq b_i \qquad \forall i \in V,\ \forall k \in K \tag{6}$$

$$x_{ijk}(t_{ik} + \tau_{ij}) \leq t_{jk} \qquad \forall (i,j) \in E,\ \forall k \in K \tag{7}$$

$$x_{ijk} \in \{0,1\} \qquad \forall (i,j) \in E,\ \forall k \in K \tag{8}$$

Here (2) ensures that every customer $i \in C$ is visited, while (3) ensures that each route starts and ends in the depot. Constraint (4) maintains flow conservation, while (5) ensures that the capacity of each vehicle is not exceeded. Constraints (6), (7) ensure that the time windows are satisfied. Note that (7) together with the assumption that $\tau_{ij} > 0$ for all $(i,j) \in E$ eliminates sub-tours. The last constraints define the domain of the arc flow variables. Note that a zero-cost edge $x_{oo'k}$ between the start and end depot must be present for all vehicles for (3) to hold if not all vehicles are used.

The standard Dantzig-Wolfe decomposition of the VRPTW, see e.g. Desrochers et al. [12], leads to the following master problem:

$$\min \sum_{p \in P} \sum_{(i,j) \in E} c_{ij} \alpha_{ijp} \lambda_p \tag{9}$$

$$\text{s.t} \sum_{p \in P} \sum_{(i,j) \in \delta^+(i)} \alpha_{ijp} \lambda_p = 1 \qquad \forall i \in C \tag{10}$$

$$\lambda_p \in \{0,1\} \qquad \forall p \in P \tag{11}$$

where P is the set of all feasible routes, the binary constant α_{ijp} is one if and only if edge (i,j) is used by route $p \in P$, and the binary variable λ_p indicates whether route p is used. The master problem can be recognized as a set partitioning problem, and the LP relaxation may be solved using delayed column generation. Let $\pi \in \mathbb{R}$ be the dual variables of (10) and let $\pi_0 = 0$. Then the reduced cost of a route p is:

$$\overline{c}_p = \sum_{(i,j) \in E} c_{ij} \alpha_{ijp} - \sum_{(i,j) \in E} \pi_j \alpha_{ijp} = \sum_{(i,j) \in E} (c_{ij} - \pi_j) \alpha_{ijp} \tag{12}$$

The pricing problem becomes an ESPPRC where the cost of each edge is $\overline{c}_{ij} = c_{ij} - \pi_j$ for all edges $(i,j) \in E$. When applying cuts during column generation we will distinguish between valid inequalities for the VRPTW constraints (2)-(8) and valid inequalities for the set partitioning constraints (10)-(11).

Consider a valid inequality for the VRPTW constraints (2)–(8) in terms of the arc flow variables x:

$$\sum_{k \in K} \sum_{(i,j) \in E} \beta_{ij} x_{ijk} \leq \beta_0 \tag{13}$$

When decomposed into the master problem, inequality (13) is reformulated as:

$$\sum_{p \in P} \sum_{(i,j) \in E} \beta_{ij} \alpha_{ijp} \lambda_p \leq \beta_0 \tag{14}$$

Let $\mu \leq 0$ be the dual variable of (14). The reduced cost of a column p is then

$$\begin{aligned} \overline{c}_p &= \sum_{(i,j) \in E} c_{ij} \alpha_{ijp} - \sum_{(i,j) \in E} \pi_j \alpha_{ijp} - \mu \sum_{(i,j) \in E} \beta_{ij} \alpha_{ijp} \\ &= \sum_{(i,j) \in E} (c_{ij} - \pi_j - \mu \beta_{ij}) \alpha_{ijp} \end{aligned} \tag{15}$$

Compared to (12) an additional coefficient $\mu \beta_{ij}$ is subtracted from the cost of edge (i,j) and the complexity of the pricing problem remains unchanged if we use the edge costs $\overline{c}_{ij} = c_{ij} - \pi_j - \mu \beta_{ij}$.

Now, consider adding a valid inequality for the set partitioning master problem (10)–(11) that cannot be written as a linear combination of the arc flow variables:

$$\sum_{p \in P} \beta_p \lambda_p \leq \beta_0 \tag{16}$$

Let $\sigma \leq 0$ be the dual variable of (16). The reduced cost of a column p is:

$$\hat{c}_p = \overline{c}_p - \sigma\beta_p = \sum_{(i,j) \in E} \overline{c}_{ij} \alpha_{ijp} - \sigma\beta_p \tag{17}$$

In addition to the reduced cost computed for a column p in (15) the cost $-\sigma\beta_p$ must be considered. To reflect the possible extra cost $-\sigma\beta_p$ it may be necessary to modify the pricing problem by adding constraints or variables, thereby increasing its complexity.

3 Subset-Row Inequalities

The set of valid inequalities for the set packing problem is a subset of the set of valid inequalities for the set partitioning problem since the latter problem is a special case of first-mentioned. Two well-known valid inequalities for the set packing problem are the clique and the odd-hole inequalities, where the first is known to be facet-defining for the set partitioning problem (Nemhauser and Wolsey [29]).

Since the master problem is a set partitioning problem, it would be obvious to go in this direction when looking for valid inequalities for the master problem. Consider the separation of a clique or an odd-hole inequality. The undirected conflict graph $G'(P, E')$ is defined as follows: Each column is a vertex in G' and the edge set is given as:

$$E' = \left\{ (p,q) : \sum_{(i,j) \in \delta^+(i)} \alpha_{ijp} = 1 \wedge \sum_{(i,j) \in \delta^+(i)} \alpha_{ijq} = 1,\ i \in C,\ p,q \in P,\ p \neq q \right\}$$

That is, an edge is present if the two columns p and q have coefficient one in the same row. In a VRPTW context it reads: Two routes are conflicting if they are visiting the same customer. A clique in G' leads to the valid clique inequality:

$$\sum_{p \in \hat{P}} \lambda_p \leq 1 \tag{18}$$

where $\hat{P} \subseteq P$ are the columns corresponding to the vertices of a clique in G'. A cycle visiting an odd number of vertices P in G' leads to the valid odd-hole inequality:

$$\sum_{p \in \hat{P}} \lambda_p \leq \left\lfloor \frac{|\hat{P}|}{2} \right\rfloor \tag{19}$$

where $\hat{P} \subseteq P$ are the columns corresponding to the vertices visited on the cycle in G'. However, when column generation is applied, it is not obvious how to reflect the reduced cost of (18) or (19) in the pricing problem since there is no specific knowledge of the columns of the master problem when solving the pricing problem.

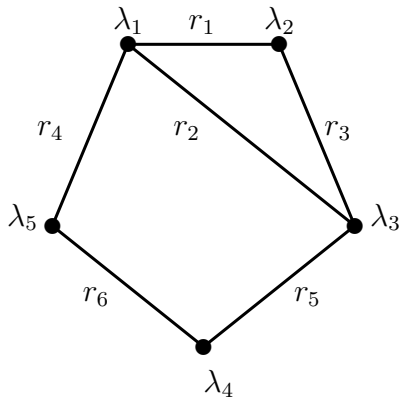
Example 1

SR inequalities derived from the conflict graph of a set packing problem. In the LP-solution to $A\lambda \leq 1$ all λ variables are $\frac{1}{2}$, which results in two violated SR inequalities:

- With $|S| = 3$ and $k = 2$ due to variables λ_1, λ_2, and λ_3 giving the set of rows $S = \{r_1, r_2, r_3\}$
- With $n = 5$ and $k = 2$ due to variables λ_1, λ_2, λ_3, λ_4, and λ_5 giving the set of rows $S = \{r_1, r_3, r_4, r_5, r_6\}$

	λ_1	λ_2	λ_3	λ_4	λ_5		
r_1	1	1				$\leq$	1
r_2	1		1			$\leq$	1
r_3		1	1			$\leq$	1
r_4	1				1	$\leq$	1
r_5			1	1		$\leq$	1
r_6				1	1	$\leq$	1

Set packing problem $A\lambda \leq 1$.



Corresponding conflict graph.

Inspired by the above inequalities (18) and (19) we introduce the *subset-row inequalities* (SR inequalities). These inequalities are specifically linked to the rows (rather than the columns) of the set packing problem, hence making it possible to identify the coefficient of a column in an SR inequality.

Definition 1. *Consider the set packing structure*

$$X = \{\lambda \in \mathbb{B}^{|P|} : A\lambda \leq 1\} \tag{20}$$

with the set of rows M and columns P, and a $|M| \times |P|$ binary coefficient matrix A. The SR inequality is defined as:

$$\sum_{p \in P} \left\lfloor \frac{1}{k} \sum_{i \in S} \alpha_{ip} \right\rfloor \lambda_p \leq \left\lfloor \frac{|S|}{k} \right\rfloor \tag{21}$$

where $S \subseteq M$ and $0 < k \leq |S|$.

Example 1 illustrates some SR inequalities derived from the conflict graph of a set packing problem.

Given a column $p \in P$ we need to have $\sum_{i \in S} \alpha_{ip} \geq k$ to get a non-zero coefficient of λ_p in (21). For the master problem of VRPTW the coefficient matrix can be translated as $\alpha_{ip} = \sum_{(i,j) \in \delta^+(i)} \alpha_{ijp}$, i.e., α_{ip} is the sum of all the outgoing edges of a customer i. Hence,

$$\left\lfloor \frac{1}{k} \sum_{i \in S} \alpha_{ip} \right\rfloor = \left\lfloor \frac{1}{k} \sum_{i \in S} \sum_{(i,j) \in \delta^+(i)} \alpha_{ijp} \right\rfloor$$

which is only 1 or larger when k or more customers of S are visited on route p.

Proposition 1. *The SR inequalities* (21) *are valid for the Set Packing structure* X*.*

Proof. The proof follows directly from Chavtal-Gomory's procedure to construct valid inequalities (Wolsey [37]). Scale the $|S|$ inequalities $\sum_{p\in P} \alpha_{ip}\lambda_p \leq 1$ for each row $i \in S \subseteq M$ from (20) with $\frac{1}{k} \geq 0$ and add them:

$$\sum_{p\in P} \frac{1}{k} \sum_{i\in S} \alpha_{ip}\lambda_p \leq \frac{|S|}{k}$$

Flooring on left side and right side leads to (21). □

Observe that, when the coefficient $\left\lfloor \frac{1}{k} \sum_{i\in S} \alpha_{ip} \right\rfloor$ evaluates to 0 or 1 for all $p \in P$ and the right hand side $\left\lfloor \frac{|S|}{k} \right\rfloor = 1$ then the set of SR inequalities (21) is a subset of the clique inequalities (18).

From Definition 1 it is clear that the SR inequalities are Chvatal-Gomory rank-1 cuts, see Chvatal [6]. Eisenbrand [15] has shown that the separation problem is $\mathcal{NP}$-complete for general Chvatal-Gomory rank-1 cuts. However, in some special cases polynomial time separation is possible, e.g. the maximally violated mod-k cuts for a fixed k by Caprara et al. [4]. Since the SR inequalities are another special case, the separation problem will be investigated further.

3.1 Separation of Subset-Row Inequalities

The separation problem of SR inequalities is defined as follows: Given the current LP-solution λ where $\lambda_p < 1$ for all $p \in P$, and let n be the size of S. For some fixed values n and k where $1 < k \leq n$, find the most violated SR inequality. Using the binary variable x_i to denote whether $i \in S$ this can be stated as:

$$\max \sum_{p\in P} \left\lfloor \frac{1}{k} \sum_{i\in M} a_{ip}x_i \right\rfloor \lambda_p - \left\lfloor \frac{n}{k} \right\rfloor \tag{22}$$

$$\text{s.t.} \sum_{i\in M} x_i = n \tag{23}$$

$$x_i \in \{0,1\} \qquad \forall i \in M \tag{24}$$

The corresponding decision problem SR-DECISION asks whether

$$\sum_{p\in P} \left\lfloor \frac{1}{k} \sum_{i\in M} a_{ip}x_i \right\rfloor \lambda_p \geq c \tag{25}$$

is feasible subject to (23) and (24), where $1 \leq c < n$ and $c \in \mathbb{Z}$. Since we may multiply (25) by any coefficient $\frac{1}{\gamma} > 0$, the coefficient bounds $\lambda_p < 1$ and $c < n$ can be softened to

$$\lambda_p < \frac{1}{\gamma}, \qquad c < \frac{n}{\gamma} \tag{26}$$

This leads to the following proposition:

Proposition 2. *The separation problem SR-DECISION is* $\mathcal{NP}$*-complete.*

Example 2

Illustration of the transformation 3CNF-SAT to SR-DECISION. Given the 3CNF-SAT expression

$$\phi = (x_1 \vee \neg x_1 \vee \neg x_2) \wedge (x_3 \vee x_2 \vee x_4) \wedge (\neg x_1 \vee \neg x_3 \vee \neg x_4)$$

the matrix $A = (a_{ij})$ becomes

		1	...	m	$m+1$	...	...	$m+n$	$m+n+1$
		C_1	...	C_m	x_1	...	...	x_n	
1	x_1	1			1				
2	$\neg x_1$	1		1	1				
	x_2		1			1			
$\vdots$	$\neg x_2$	1				1			
	x_3		1				1		
	$\neg x_3$			1			1		
	x_4		1					1	
$2n$	$\neg x_4$			1				1	
$2n+1$									1
$2n+2$		1	1	1	1	1	1	1	1
$2n+3$		1	1	1	1	1	1	1	1

while we set $k = 3$, $\lambda_p = 1$ for $p \in P$ and $c = 8$.

Proof. We will show the statement by reduction from 3-conjunctive normal form satisfiability (3CNF-SAT). Given an expression ϕ written in three-conjunctive normal form, the 3CNF-SAT problem asks whether there is an assignment of binary values to the variables such that ϕ evaluates to true. An expression is in three-conjunctive normal form when it consists of a collection of disjunctive clauses $C_1, \ldots, C_m$ of literals, where a literal is a variable x_i or a negated variable $\neg x_i$, and each clause contains exactly three literals.

Let $x_1, \ldots, x_n$ be the set of variables which occurs in the clause ϕ. We transform the 3CNF-SAT instance to a SR-DECISION instance by constructing a matrix $A = (a_{ij})$ with $2n + 3$ rows and $m + n + 1$ columns, i.e., $M = \{1, \ldots, 2n + 3\}$ and $P = \{1, \ldots, m + n + 1\}$.

The rows $1, \ldots, 2n$ of matrix A corresponds to literals $x_1, \neg x_1, x_2, \neg x_2, \ldots, x_n, \neg x_n$, while columns $j = 1, \ldots, m$ correspond to clauses $C_1, \ldots, C_m$, and columns $j = m + 1, \ldots, m + n$ correspond to variables $x_1, \ldots, x_n$.

We now define matrix A as follows: For $j = 1, \ldots, m$ let $a_{ij} = 1$ iff the corresponding literal appears in clause C_j. For $j = 1, \ldots, n$ let $a_{i,j+m} = 1$ iff the corresponding literal is x_j or $\neg x_j$. For $j = m + n + 1$ let $a_{ij} = 0$. The last three rows of A are defined as follows: For $j = 1, \ldots, m + n$ let $a_{2n+1,j} = 0$, while $a_{2n+1,m+n+1} = 1$. For $j = 1, \ldots, m + n + 1$ let $a_{2n+2,j} = a_{2n+3,j} = 1$. Finally we set $k = 3$, $\lambda_p = 1$ for all $p \in P$ and $c = m + n + 1$. Note that all coefficients are within the bounds (26) for γ sufficiently large. An example of the transformation is illustrated in Example 2.

With the chosen constants, the SR-DECISION problem (25) reads

$$\sum_{p \in P} \left\lfloor \frac{1}{3} \sum_{i \in M} a_{ip} x_i \right\rfloor \geq m + n + 1 = |P|$$

which is satisfied if and only if

$$\sum_{i \in M} a_{ip} x_i \geq 3 \qquad \forall p \in P$$

As the last three rows of A always must be chosen, it is equivalent to

$$\sum_{i=1}^{2n} a_{ip} x_i \geq 1 \qquad \forall p = 1, \ldots, m+n$$

(i) Assume that there is a feasible assignment of binary values to $x_1, \ldots, x_n$ such that ϕ evaluates to true in the 3CNF-SAT instance. In the corresponding SR-DECISION problem choose row i if and only if the corresponding literal is true in ϕ. Since exactly n literals are true, we will in this way choose n rows. Since at least one literal is true in each clause, and each column $1, \ldots, m$ corresponds to a clause in A we will get a contribution of at least one in each of these columns. Moreover, since exactly one of x_i and $\neg x_i$ is true in ϕ we will get a contribution of exactly one in column $m+1, \ldots, m+n$. Hence, the corresponding SR-DECISION problem is true.

(ii) Assume on the other hand that SR-DECISION is true. Let $P' \subseteq P$ be the set of rows corresponding to the solution. By assumption $|P'| = n$. First we notice that exactly one of the rows corresponding to the literals x_i and $\neg x_i$ is chosen. This follows from the fact that we have n columns $m+1, \ldots, m+n$ which needs to be covered by n rows, and each row covers exactly one column. For each literal in ϕ let x_i or $\neg x_i$ be true if the corresponding row was chosen in SR-DECISION. Each variable will be well-defined due to the above argument. Moreover, since the rows P' must cover at least one $a_{pi} = 1$ for each column $j = 1, \ldots, m$, we see that each clause in ϕ becomes true.

Since the reduction is polynomial, and SR-DECISION obviously is in $\mathcal{NP}$, we have proved the statement. □

Example 3 shows that typical separation problems of SR inequalities actually possess the properties assumed in the $\mathcal{NP}$-completeness proof.

4 Label-Setting Algorithm

When solving the pricing problem, it is noted that finding a route with negative reduced cost corresponds to finding a negative cost path starting and ending at the depot, i.e., an ESPPRC. Our ESPPRC algorithm is based on standard label setting techniques presented by e.g. Beasley and Christofides [2], Dumitrescu [14], Feillet et al. [16], Chabrier [5], Danna and Le Pape [9]; hence in the following we mainly focus on the dominance criterion used for handling the modifications stemming from the SR inequalities of the master problem.

The ESPPRC can be formally defined as: Given a weighted directed graph $G(V, E)$ with nodes V and edges E, and a set of resources R. For each edge $(i, j) \in E$ and resource $r \in R$ three parameters are given: A lower limit $a_r(i, j)$ on the accumulation of resource r when traversing edge $(i, j) \in E$; an upper limit $b_r(i, j)$ on the accumulation of resource r when traversing edge $(i, j) \in E$; and finally an amount $c_r(i, j)$ of resource r consumed by traversing edge $(i, j) \in E$. The objective is to find a minimum cost path p from a source node $o \in V$ to

Example 3

To illustrate that the bounds (26) indeed are realistic consider the case $k = 3$. Choose $\gamma = \frac{m+n+1}{\beta}$ where $\beta = \frac{n-2}{3}$ or $\beta = \frac{n-1}{3}$ depending on which of the expressions that evaluates to an integral value. The right hand side of (25) evaluates to

$$c \cdot \frac{1}{\gamma} = (m+n+1) \cdot \frac{\beta}{m+n+1} = \beta$$

where an integral value of β gives

$$\beta = \left\lfloor \frac{n}{3} \right\rfloor < n$$

The value of λ gives

$$\lambda_p \cdot \frac{1}{\gamma} = 1 \cdot \frac{\beta}{m+n+1} \leq 1 \qquad \forall p \in P$$

Hence all bounds are valid according to the separation problem (22)-(24).

a target node $o' \in V$, where the accumulated resources of p satisfy the limits for all resources $r \in R$. Without loss of generality, we assume that the limits must be satisfied at the start of each edge (i, j), i.e., before $c_r(i, j)$ has been consumed.

Remark that equivalent upper and lower limits and consumptions on the nodes can be "pushed" onto the edges, e.g., the ingoing edges of the node.

For the pricing problem of the VRPTW, the resources are demand d, time t, a binary visit-counter for each customer $v \in C$ and reduced cost $\overline{c}$. Note that also the reduced cost is considered a resource. When considering the pricing problem of the VRPTW, the consumptions and upper and lower limits of the resources at each edge (i, j) in ESPPRC are:

$$\begin{array}{llll}
a_d(i,j) = 0, & b_d(i,j) = D - d_j, & c_d(i,j) = d_j & \forall (i,j) \in E \\
a_t(i,j) = a_i, & b_t(i,j) = b_i, & c_t(i,j) = \tau_{ij} & \forall (i,j) \in E \\
a_v(i,j) = 0, & b_v(i,j) = 1, & c_v(i,j) = 1 \quad \forall v \in V : v = j, & \forall (i,j) \in E \\
a_v(i,j) = 0, & b_v(i,j) = 1, & c_v(i,j) = 0 \quad \forall v \in V : v \neq j, & \forall (i,j) \in E \\
a_{\overline{c}}(i,j) = -\infty, & b_{\overline{c}}(i,j) = \infty\ , & c_{\overline{c}}(i,j) = \overline{c}_{ij} & \forall (i,j) \in E
\end{array}$$

In the label-setting algorithm labels at node v represent partial paths from o to v. The following attributes for a label L are considered:

- $\overline{v}(L)$ The current end-node of the partial path represented by L.
- $\overline{c}(L)$ The sum of the reduced cost along path L.
- $r(L)$ The accumulated consumption of resource $r \in R$ along path L.

A feasible extension $\epsilon \in \mathcal{E}(L)$ of a label L is a partial path starting in a node $\overline{v}(L) \in V$ and ending in the target node o', that does not violate any resources when concatenated with the partial path represented by L.

In the following it is assumed that all resources are bounded strongly from above, and weakly from below. This means that if the current resource accumulation of a label is below the lower limit on a given edge, it is allowed to fill up the resource to the lower limit, e.g., waiting for a time window to open. This means that two consecutive labels L_u and L_v related by an edge (u, v), i.e., L_u is extended and creates L_v, where $\overline{v}(L_u) = u$ and $\overline{v}(L_v) = v$, must

satisfy

$$r(L_v) \leq b_r(u,v), \qquad \forall r \in R \tag{27}$$
$$r(L_v) = \max\{r(L_u) + c_r(u,v), a_r(u,v)\}, \qquad \forall r \in R \tag{28}$$

Here (27) demands that each label L_u satisfies the upper limit $b_r(u,v)$ of resource r corresponding to edge (u,v), while (28) states that resource r at label L_v corresponds to the resource consumption at label L_u plus the amount consumed by traversing edge (u,v), respecting the lower limit $a_r(u,v)$ on edge (u,v).

A simple enumeration algorithm could be used to produce all these labels, but to limit the number of labels considered, dominance rules are introduced to fathom labels which do not lead to an optimal solution.

Definition 2. *A label L_i dominates label L_j if*

$$\overline{v}(L_i) = \overline{v}(L_j) \tag{29}$$
$$\overline{c}(L_i) \leq \overline{c}(L_j) \tag{30}$$
$$\mathcal{E}(L_j) \subseteq \mathcal{E}(L_i) \tag{31}$$

In other words, the paths corresponding to labels L_i and L_j should end at the same node $\overline{v}(L_i) = \overline{v}(L_j) \in V$, the path corresponding to label L_i should cost no more than the path corresponding to label L_j, and finally any feasible extension of L_j is also a feasible extension of L_i.

Feillet et al. [16] suggested to consider the set of nodes that cannot be reached from a label L_i and compare the set with the unreachable nodes of a label L_j in order to determine if some extensions are impossible. Or in other words: update the node resources in an eager fashion instead of a lazy. The following definition is a generalization of Definition 3 in Feillet et al. [16].

Definition 3. *Given a start node $o \in V$, a label L, and a node $u \in V$ where $\overline{v}(L) = u$ a node $v \in V$ is considered* unreachable *if v has already been visited on the path from o to u or if a resource window is violated, e.g.:*

$$\exists r \in R \qquad\qquad r(L) + \ell_r(u,v) > b_r(v)$$

where $\ell_r(u,v)$ is a lower bound on the consumption of resource r on all feasible paths from u to v. The node resources *are then given as: $v(L) = 1$ indicates that node $v \in V$ is unreachable from node $\overline{v}(L) \in V$, and $v(L) = 0$ otherwise.*

Determining if (31) holds can be quite cumbersome because the straightforward definition demands that we calculate all extensions of the two labels. Therefore, a sufficient criterion for (31) is sought that can be computed faster. If label L_i has consumed less resources than label L_j then no resources are limiting the possibilities of extending L_i compared to L_j, hence the following proposition can be used as a relaxed version of the dominance criterion.

Proposition 3. Desaulniers et al. [10]. *If all resource extension functions are non-decreasing, then label L_i dominates label L_j if:*

$$\overline{v}(L_i) = \overline{v}(L_j) \tag{32}$$
$$\overline{c}(L_i) \leq \overline{c}(L_j) \tag{33}$$
$$r(L_i) \leq r(L_j) \qquad \forall r \in R \tag{34}$$

Using Proposition 3 as a dominance criterion is a relaxation of the dominance criterion of Definition 2 since only a subset of labels satisfying (29), (30) and (31) satisfies inequalities (32), (33) and (34).

4.1 Solving the Modified Pricing Problem

Consider some valid SR inequality of the form (21),

$$\sum_{p \in P} \left\lfloor \frac{1}{k} \sum_{i \in S} \alpha_{ip} \right\rfloor \lambda_p \leq \left\lfloor \frac{|S|}{k} \right\rfloor$$

where $S \subseteq M$ and $0 < k \leq |S|$. Let $\sigma \leq 0$ be the corresponding dual variable when solving the master problem to LP-optimality. From (17) the reduced cost of a column in the VRPTW master problem is:

$$\hat{c}_p = \overline{c}_p - \sigma \left\lfloor \frac{\sum_{i \in S} \sum_{(i,j) \in \delta^+(i)} \alpha_{ijp}}{k} \right\rfloor = \sum_{(i,j) \in E} \overline{c}_{ij} \alpha_{ijp} - \sigma \left\lfloor \frac{\sum_{i \in S} \sum_{(i,j) \in \delta^+(i)} \alpha_{ijp}}{k} \right\rfloor \tag{35}$$

We analyze how this additional cost can be handled in the label-setting algorithm for ESPPRC.

Let $V(L)$ be the nodes visited on the partial path of label L. The cost of a label L can then be expressed as:

$$\hat{c}(L) = \overline{c}(L) - \sigma \left\lfloor \frac{|S \cap V(L)|}{k} \right\rfloor \tag{36}$$

A new resource m can be used to compute the coefficient of penalty σ for label L, i.e., $m(L) = |S \cap V(L)|$, the number of customers involved in the cut. Note that the consumption of resource m is 1 for each e.g. outgoing edge of the involved customers. Therefore the usual dominance criterion of Proposition 3 can be used. Note that in case L_i dominates L_j, $\overline{c}(L_i) \leq \overline{c}(L_j)$ and $m(L_i) \leq m(L_j)$ so $\hat{c}(L_i) \leq \hat{c}(L_j)$ since $-\sigma > 0$. Hence the penalty term must only be considered on the last edge to the target node to compute the reduced cost $\hat{c}(L)$ of path L. However, further labels can be eliminated by exploiting the structure of (36).

For a label L let

$$\mathcal{T}(L) = |S \cap V(L)| \bmod k$$

be the number of visits made to S since the last penalty was paid for visiting k nodes in S. Recall $\mathcal{E}(L)$ as the set of feasible extensions from the label L to the target node o' and note that when label L_i dominates label L_j, their common extensions are $\mathcal{E}(L_j)$ due to (31). The following cost dominance criterion is obtained for a single SR inequality:

Proposition 4. *If $\mathcal{T}(L_i) \leq \mathcal{T}(L_j)$, $\overline{v}(L_i) = \overline{v}(L_j)$, $\hat{c}(L_i) \leq \hat{c}(L_j)$, and $r(L_i) \leq r(L_j)$ $\forall r \in R$, then label L_i dominates label L_j.*

Proof. Consider any common extension $\epsilon \in \mathcal{E}(L_j)$. Since $\mathcal{T}(L_i) \leq \mathcal{T}(L_j)$ the relation between the number of future penalties for the two labels when concatenated with ϵ is:

$$\left\lfloor \frac{|S \cap \epsilon| + \mathcal{T}(L_i)}{k} \right\rfloor \leq \left\lfloor \frac{|S \cap \epsilon| + \mathcal{T}(L_j)}{k} \right\rfloor$$

This leads to the following relation between the costs:

$$\begin{aligned} & \hat{c}(L_i + \epsilon) = \hat{c}(L_i) + \overline{c}(\epsilon) - \sigma \left\lfloor \frac{|S \cap \epsilon| + \mathcal{T}(L_i)}{k} \right\rfloor \\ \leq \quad & \hat{c}(L_j + \epsilon) = \hat{c}(L_j) + \overline{c}(\epsilon) - \sigma \left\lfloor \frac{|S \cap \epsilon| + \mathcal{T}(L_j)}{k} \right\rfloor \end{aligned}$$

Hence label L_i dominates label L_j. □

Proposition 5. *If $\mathcal{T}(L_i) > \mathcal{T}(L_j)$, $\overline{v}(L_i) = \overline{v}(L_j)$, $\hat{c}(L_i) - \sigma \leq \hat{c}(L_j)$, and $r(L_i) \leq r(L_j)$ $\forall r \in R$, then label L_i dominates label L_j.*

Proof. Consider any common extension $\epsilon \in \mathcal{E}(L_j)$. Since $\mathcal{T}(L_i) > \mathcal{T}(L_j)$ the relation between the number of future penalties for the two labels when concatenated with ϵ is:

$$\left\lfloor \frac{|S \cap \epsilon| + \mathcal{T}(L_i)}{k} \right\rfloor \geq \left\lfloor \frac{|S \cap \epsilon| + \mathcal{T}(L_j)}{k} \right\rfloor \tag{37}$$

Since $0 \leq \mathcal{T}(L_j) < \mathcal{T}(L_i) \leq k$ it is clear that the left hand side of (37) is at most one unit larger than the right hand side, i.e., label L_i will pay the penalty at most one more time than label L_j. Hence,

$$\left\lfloor \frac{|S \cap \epsilon| + \mathcal{T}(L_i)}{k} \right\rfloor - 1 \leq \left\lfloor \frac{|S \cap \epsilon| + \mathcal{T}(L_j)}{k} \right\rfloor$$

That is, the additional cost of extending L_i with ϵ is at most $-\sigma$ more than extending L_j with ϵ. This leads to the following relation between the costs:

$$\begin{aligned} \hat{c}(L_i + \epsilon) &= \hat{c}(L_i) + \overline{c}(\epsilon) - \sigma \left\lfloor \frac{|S \cap \epsilon| + \mathcal{T}(L_i)}{k} \right\rfloor \\ &= \hat{c}(L_i) - \sigma + \overline{c}(\epsilon) - \sigma \left(\left\lfloor \frac{|S \cap \epsilon| + \mathcal{T}(L_i)}{k} \right\rfloor - 1 \right) \\ &\leq \hat{c}(L_j) + \overline{c}(\epsilon) - \sigma \left\lfloor \frac{|S \cap \epsilon| + \mathcal{T}(L_j)}{k} \right\rfloor \\ &= \hat{c}(L_j + \epsilon) \end{aligned}$$

Hence label L_i dominates label L_j. □

Observe that if $\mathcal{T}(L_i) + |S \cap \epsilon| < k$ for all $\epsilon \in \mathcal{E}(L_j)$, it is not possible to visit S enough times to trigger a penalty, i.e., the temporary penalty to the cost of L_i can be disregarded.

In case of several SR inequalities, the new dominance criterion is as follows:

Proposition 6. *Let $Q = \{q : \sigma_q < 0 \wedge \mathcal{T}_q(L_i) > \mathcal{T}_q(L_j)\}$. Then label L_i dominates label L_j if:*

$$\overline{v}(L_i) = \overline{v}(L_j) \tag{38}$$

$$\hat{c}(L_i) - \sum_{q \in Q} \sigma_q \leq \hat{c}(L_j) \tag{39}$$

$$r(L_i) \leq r(L_j) \qquad \forall r \in R \tag{40}$$

Proof. The validity of (39) follows directly from Propositions 4 and 5. The validity of (38) and (40) follows from Proposition 3. □

5 Computational Results

The BCP algorithm has been implemented using the BCP framework and the open source linear programming solver CLP, both parts of the framework COIN [7]. All tests are run on an Intel® Pentium® 4 3.0 GHz PC with 4 GB of memory.

The benchmarks of Solomon [34] follow a naming convention of `DTm.n`. The distribution `D` can be R, C and RC, where the C instances have a clustered distribution of customers, the R instances have a random distribution of customers, and the RC instances are a mix of clustered and randomly distributed customers. The time window `T` is either 1 or 2, where instances of type 1 have tighter time windows than instances of type 2. The instance number is given by `m` and the number of customers is given by `n`.

The outline of the BCP algorithm presented in this paper is as follows:

Step 1. Choose an unprocessed branch node. If the lower bound is above the upper bound, then fathom branch node.

Step 2. Solve the LP master problem.

Step 3. Solve the pricing problem heuristically. If columns with negative reduced cost have been found, then add them to the master problem and go back to Step 2.

Step 4. Solve the pricing problem to optimality. Update the lower bound. If the lower bound is above the upper bound, then fathom the branch node. If some new columns have been found, then add them to the master problem and go to Step 2.

Step 5. Separate SR inequalities. If any violated cuts are found, then add them to the master problem and go to Step 2.

Step 6. If the LP solution is fractional then branch and add the children to the set of unprocessed branch nodes. Mark the current node as processed and go to Step 1.

We allow a maximum of 400 variables and 50 cuts to be generated in each of steps 3, 4, and 5 respectively. The pricing-problem heuristic is based on the label-setting algorithm but a simpler heuristic dominance criterion is used. If a label L_i dominates L_j on *cost*, *demand* and *time* it is regarded as dominated and L_j is discarded. That is, no concern is taken to the *node* resources. The separation of SR inequalities is done with a complete enumeration of all inequalities with $|S| = 3$ and $k = 2$. Let B be the set of basic variables in the current LP solution and C be the set of customers, then the separation can be done in $O(|C|^3|B|)$. Preliminary tests showed that SR inequalities with different values of n and k seldom appeared in the VRPTW instances, hence no separation of these inequalities was done.

The branch tree is explored with a best-bound search strategy, i.e., the node with the lowest lower bound is chosen first, breaking ties based on the LP result of the strong branching. We have adapted the branching rule used by Fukasawa et al. [17]: For a subset of customers $S \subset C$ the number of vehicles to visit that set is either two or greater than or equal to four, i.e.,

$$\sum_{k \in K} \sum_{(i,j) \in \delta^+(S)} (x_{ijk} + x_{jik}) = 2$$

and

$$\sum_{k \in K} \sum_{(i,j) \in \delta^+(S)} (x_{ijk} + x_{jik}) \geq 4$$

We are using the cut library of Lysgaard [25] to separate candidate sets for branching, which is an implementation of the heuristic methods described in Lysgaard et al. [26].

Author(s)	CPU	SpecINT	SpecCFP	Normalized
Irnich and Villeneuve [20]	P3 600 MHz*	295	204	0.23
Chabrier [5]	P4 1.5 GHz	526	606	0.52
Jepsen et al. [this paper]	P4 3.0 GHz	1099	1077	1.00

Table 1: Comparison of computer speed. Based on CPU2000 benchmarks from SPEC [35]. (*) benchmarks are given for P3 650 MHz since no benchmarks were available for P3 600. The normalized value is an average of SpecINT and SpecCFP.

5.1 Running Times

To give a fair comparison between running times of our algorithm and the two most recent algorithms presented by Irnich and Villeneuve [20] and Chabrier [5], the CPU speed is taken into account. This is done according to the CPU2000 benchmarks reported by The Standard Performance Evaluation Corporation SPEC [35]. Table 1 gives the integer and floating point benchmark scores and a normalized value, e.g. our computations were carried out on a computer approximately twice as fast as that of Chabrier.

A comparison of running times is shown in Table 2. To save space we only report results on what we consider hard instances, i.e., the Solomon instances that were closed by either Irnich and Villeneuve [20] or Chabrier [5] and by us.

Our algorithm outperforms those of Irnich and Villeneuve and Chabrier for 17 out of 22 instances. Seven of these instances were solved without any SR inequalities. In these cases, the faster running times were probably due to the bi-directional label-setting algorithm.

With the introduction of SR inequalities our algorithm becomes competitive with the algorithm based on solving k-cyc-SPPRC (e.g. instances R104.100, RC104.100, RC107.100, RC108.100, and R211.50) and clearly outperforms the ESPPRC based algorithm on the harder instances (e.g., instances R210.50, RC202.100, RC205.100, and RC208.25). In some cases when solving the C1 and C2 instances the BCP algorithm tails off leading to slow solution times or no solution at all. However, this must be seen in the light of a simple implementation and no use of other cutting planes than the SR inequalities.

5.2 Comparing Lower Bounds in the Root Node

Table 3 reports the lower bounds obtained in the root node of the master problem with and without SR inequalities and with best bounds obtained by Irnich and Villeneuve [20] using k-cyc-SPPRC. Again we only report results on what we consider the hard instances from Table 2 plus the instances closed by us.

As seen, the lower bounds obtained with SR inequalities are improved quite significantly for most of the instances. Moreover, in most cases the problems are solved without branching. Out of the 32 instances considered, the gap was closed in the root node in 8 instances due to the ESPPRC and in an additional 16 instances due to the SR inequalities. However, one needs to take into account that the running time of solving the root node is increased due to the increased difficulty of the pricing problems.

	Irnich and Villeneuve [20]	Chabrier [5]	Jepsen et al. [this paper]			
Instance	Time (s)	Time (s)	Time (s)	Speedup		
R104.100	268106.0	-	**32343.9**	1.9	/	-
RC104.100	986809.0	-	**65806.8**	3.4	/	-
RC107.100	42770.7	-	**153.8**	64.0	/	-
RC108.100	71263.0	-	**3365.0**	4.9	/	-
R203.50	**217.1**	3320.9	50.8	1.0	/	34.0
R204.25	123.1	171.6	**7.5**	3.8	/	11.9
R205.50	585.7	531.0	**15.5**	8.6	/	17.8
R206.50	22455.3	4656.1	**190.9**	27.1	/	12.7
R208.25	321.9	741.5	*__2.9__	25.5	/	133.0
R209.50	142.4	195.4	**16.6**	2.0	/	6.1
R210.50	11551.4	65638.6	*__332.7__	8.0	/	102.6
R211.50	**21323.0**	-	10543.8	0.5	/	-
RC202.50	241.6	**13.0**	*10.7	5.2	/	0.6
RC202.100	124018.0	19636.5	**312.6**	91.2	/	32.7
RC203.25	1876.0	5.1	*__0.7__	616.4	/	3.8
RC203.50	54229.2	4481.5	*__190.9__	65.3	/	12.2
RC204.25	-	13.0	*__2.0__	-	/	3.4
RC205.50	52.6	**10.6**	*5.9	2.1	/	0.9
RC205.100	13295.9	15151.7	**221.2**	13.8	/	35.6
RC206.50	469.1	**9.4**	*8.2	13.2	/	0.6
RC207.50	-	71.1	*__21.5__	-	/	1.7
RC208.25	-	33785.3	**78.4**	-	/	224.1

Table 2: Comparison of running time. Speedup is calculated based on the normalized values in Table 1 and are versus Irnich and Villeneuve and Chabrier respectively. Results with (*) are based on an algorithm without the SR inequalities. Results in **boldface** indicate the fastest algorithm after normalization. (-) indicates that no running times were provided by the author(s) or that the instance was not solved.

		Irnich and Villeneuve [20]		Jepsen et al. [this paper]	
Instance	UB	k	LB	LB(1)	LB(2)
R104.100	971.5	3	955.8	956.9	971.3
R108.100	932.1	4	913.9	913.6	**932.1**
R112.100	948.6	3	925.9	926.8	946.7
RC104.100	1132.3	3	1114.4	1101.9	1129.9
RC106.100	1372.7	4	1343.1	1318.8	1367.3
RC107.100	1207.8	4	1195.4	1183.4	**1207.8**
RC108.100	1114.2	3	1100.5	1073.5	**1114.2**
R202.100	1029.6	0	933.5	1022.3	1027.3
R203.50	605.3	4	598.6	598.6	**605.3**
R203.100	870.8	2	847.1	867.0	**870.8**
R204.25	355.0	4	349.1	350.5	**355.0**
R205.50	690.1	4	682.8	682.9	**690.1**
R206.50	632.4	4	621.3	626.4	**632.4**
R207.50	575.5	4	557.4	564.1	**575.5**
R208.25	328.2	4	327.1	**328.2**	**328.2**
R209.50	600.6	4	599.9	599.9	**600.6**
R209.100	854.8	3	834.4	841.5	854.4
R210.50	645.6	4	633.1	636.1	645.3
R211.50	535.5	4	526.0	528.7	**535.5**
RC202.50	613.6	4	604.5	**613.6**	**613.6**
RC202.100	1092.3	3	1055.0	1088.1	**1092.3**
RC203.25	326.9	4	297.7	**326.9**	**326.9**
RC203.50	555.3	4	530.0	**555.3**	**555.3**
RC203.100	923.7	0	693.7	922.6	**923.7**
RC204.25	299.7	4	266.3	**299.7**	**299.7**
RC205.50	630.2	4	**630.2**	**630.2**	**630.2**
RC205.100	1154.0	3	1130.5	1147.7	**1154.0**
RC206.50	610.0	4	597.1	**610.0**	**610.0**
RC206.100	1051.1	3	1017.0	1038.6	**1051.1**
RC207.50	558.6	4	504.9	**558.6**	**558.6**
RC208.25	269.1	4	238.3	**269.1**	**269.1**
RC208.50	476.7	3	422.3	472.3	**476.7**

Table 3: Comparison of root lower bounds. LB by Irnich and Villeneuve is the best lower bound obtained with k-cyc-SPPRC and valid inequalities, LB(1) is with ESPPRC and LB(2) is with ESPPRC and SR inequalities. Lower bounds in **boldface** indicate lower bounds equal to the upper bound. Instances in **boldface** are the Solomon instances closed by us.

		25 customers		50 customers		100 customers	
Class	No.	Prev.	Jepsen et al. [this paper]	Prev.	Jepsen et al. [this paper]	Prev	Jepsen et al. [this paper]
R1	12	12	12	12	12	10	12
C1	9	9	9	9	9	9	9
RC1	8	8	8	8	8	8	8
R2	11	11	11	9	9	1	4
C2	8	8	8	8	7	8	7
RC2	8	8	8	8	7	3	5
Summary	56	56	56	55	52	39	45

Table 4: Summary of solved Solomon instances. No. is the number of instances in that class, and for 25, 50 and 100 customers the two columns refers to the number of instances previously solved to optimality and the number of instances solved to optimality by us.

Instance	UB	LB	Vehicles	Tree	LP	$\text{Time}_{\text{root}}$(s)	Time_{var}(s)	Time_{LP}(s)	Time (s)
R108.100	932.1	932.1	10	1	132	5911.71	5796.04	77.36	5911.74
R112.100	948.6	946.7	10	9	351	55573.68	199907.03	1598.63	202803.94
R202.100	1029.6	1027.3	8	13	514	974.51	730.04	4810.47	8282.38
R203.100	870.8	870.8	6	1	447	54187.15	48474.45	3973.42	54187.40
R207.50	575.5	575.5	3	1	107	34406.92	34282.47	118.69	34406.96
R209.100	854.8	854.4	5	3	337	31547.45	74779.58	2978.42	78560.47
RC203.100	923.7	923.7	5	1	402	14917.18	13873.53	1025.65	14917.36
RC206.100	1051.1	1051.1	7	1	179	339.63	159.33	171.34	339.69

Table 5: Instances closed by Jepsen et al. [this paper]. *UB* is the optimal solution found by us, *LB* is lower bound at the root node, *Vehicles* is the number of vehicles in the solution, *Tree* is the number of branch nodes, *LP* is the number of LP iterations, $Time_{root}$ is the time solving the root node, $Time_{var}$ is time spent solving the pricing problem, $Time_{LP}$ is the time spent solving LP problems, and *Time* is the total time.

5.3 Closed Solomon Instances

Table 4 gives an overview of how many instances were solved for each class of the Solomon instances. We were able to close 8 previously unsolved instances. We did not succeed to solve four previously solved instances (R204.50, C204.50, C204.100, and RC204.50).

Information on all solved Solomon instances can be found in Tables 6–8 in Appendix A. Furthermore Table 5 provides detailed information of the instances closed in this paper. The solutions can be found in Tables 9–16 in Appendix B.

6 Concluding Remarks

The introduction of the SR inequalities significantly improved the results of the BCP algorithm. This made it possible to solve 8 previously unsolved instances from the Solomon benchmarks.

Except for four cases (R204.50, C204.50 and C204.100 solved with k-cyc-SPPRC by Irnich and Villeneuve [20] and RC204.50 solved by Danna and Le Pape [9]) our BCP algorithm is competitive and in most cases superior to earlier algorithms within this field. With minor modifications in the definition of the conflict graph the SR inequalities can be applied to the k-cyc-SPPRC algorithm using the same cost-modified dominance criterion as described in this paper. Preliminary results by Jepsen et al. [21] have shown that the lower bounds obtained in a BCP algorithm for VRPTW using the k-cyc-SPPRC algorithm and SR inequalities are almost as good as those obtained using the approach presented in this paper. This seems to be a promising direction of research in order to solve large VRPTW instances, since the ESPPRC algorithm is considerably slower than the k-cyc-SPPRC algorithm when the number of customers increases.

Moreover, we note that the SR inequalities can be applied to any set packing problem. That is, they can be used in BCP algorithms for other problems with a set packing problem master problem. One only needs to consider how the dual variables of the SR inequalities are handled in the pricing problems, however this is not necessarily trivial and must be investigated for the individual pricing problems.

Adding SR inequalities to the master problem means that the pricing problem becomes a shortest path problem with non-additive non-decreasing constraints or objective function. By modifying the dominance criterion, we have shown that this is tractable in a label-setting algorithm. A further discussion of shortest path problems with various non-additive constraints can be found in Pisinger and Reinhardt [30]. The development of algorithms which efficiently handle non-additive constraints is important to increase the number of valid inequalities which can be handled.

A Results on Solomon Instances

This appendix contains detailed information about solved Solomon instances. The first column of the tables is the instance name, then three columns for the branch-and-cut-and-price algorithm with ESPPRC and with ESPPRC and SR-inequalities follow. The columns are the lower bound in the root node, the number of branch tree nodes and the total running time. A (-) means that the instance was not solved. The last two columns are the optimal upper bound and a reference to the authors who were the first to solve that instance, disregarding Desrochers et al. [12] who solved many of the instances with a different calculation of the travel times making it hard to compare with later solutions. The author legend is:

C:	Chabrier [5]
CR:	Cook and Rich [8]
DLP:	Danna and Le Pape [9]
IV:	Irnich and Villeneuve [20]
JPSP:	Jepsen et al. [this paper]
KDMSS:	Kohl et al. [23]
KLM:	Kallehauge et al. [22]
L:	Larsen [24]
S:	Salani [33]

	with ESPPRC			with ESPPRC and SR				
Instance	LB	Tree	Time (s)	LB	Tree	Time (s)	UB	Ref.
R101	617.1	1	0.02	617.1	1	0.02	617.1	KDMSS
R102	546.4	3	0.13	547.1	1	0.09	547.1	KDMSS
R103	454.6	1	0.11	454.6	1	0.11	454.6	KDMSS
R104	416.9	1	0.12	416.9	1	0.12	416.9	KDMSS
R105	530.5	1	0.02	530.5	1	0.02	530.5	KDMSS
R106	457.3	5	0.29	465.4	1	0.10	465.4	KDMSS
R107	424.3	1	0.12	424.3	1	0.12	424.3	KDMSS
R108	396.9	3	0.31	397.3	1	0.24	397.3	KDMSS
R109	441.3	1	0.06	441.3	1	0.06	441.3	KDMSS
R110	438.4	17	1.16	444.1	3	0.29	444.1	KDMSS
R111	427.3	3	0.23	428.8	1	0.13	428.8	KDMSS
R112	387.1	13	1.19	393.0	1	0.52	393.0	KDMSS
C101	191.3	1	0.13	191.3	1	0.13	191.3	KDMSS
C102	190.3	1	0.53	190.3	1	0.53	190.3	KDMSS
C103	190.3	1	0.80	190.3	1	0.80	190.3	KDMSS
C104	186.9	1	3.29	186.9	1	3.29	186.9	KDMSS
C105	191.3	1	0.17	191.3	1	0.17	191.3	KDMSS
C106	191.3	1	0.14	191.3	1	0.14	191.3	KDMSS
C107	191.3	1	0.20	191.3	1	0.20	191.3	KDMSS
C108	191.3	1	0.37	191.3	1	0.37	191.3	KDMSS
C109	191.3	1	0.62	191.3	1	0.62	191.3	KDMSS
RC101	406.7	5	0.20	461.1	1	0.09	461.1	KDMSS
RC102	351.8	1	0.05	351.8	1	0.05	351.8	KDMSS
RC103	332.8	1	0.19	332.8	1	0.19	332.8	KDMSS
RC104	306.6	1	0.52	306.6	1	0.52	306.6	KDMSS
RC105	411.3	1	0.06	411.3	1	0.06	411.3	KDMSS
RC106	345.5	1	0.10	345.5	1	0.10	345.5	KDMSS
RC107	298.3	1	0.29	298.3	1	0.29	298.3	KDMSS
RC108	294.5	1	0.67	294.5	1	0.67	294.5	KDMSS
R201	460.1	3	0.44	463.3	1	0.27	463.3	CR+KLM
R202	410.5	1	0.61	410.5	1	0.61	410.5	CR+KLM
R203	391.4	1	0.80	391.4	1	0.80	391.4	CR+KLM
R204	350.5	19	18.40	355.0	1	7.51	355.0	IV+C
R205	390.6	3	1.62	393.0	1	1.06	393.0	CR+KLM
R206	373.6	3	1.67	374.4	1	0.93	374.4	CR+KLM
R207	360.1	5	4.03	361.6	1	1.39	361.6	KLM
R208	328.2	1	2.87	328.2	1	2.87	328.2	IV+C
R209	364.1	9	4.99	370.7	1	2.26	370.7	KLM
R210	404.2	3	1.52	404.6	1	1.04	404.6	CR+KLM
R211	341.4	29	38.17	350.9	1	22.62	350.9	KLM
C201	214.7	1	0.84	214.7	1	0.84	214.7	CR+L
C202	214.7	1	3.00	214.7	1	3.00	214.7	CR+L
C203	214.7	1	3.02	214.7	1	3.02	214.7	CR+L
C204	213.1	1	7.00	213.1	1	7.00	213.1	CR+KLM
C205	214.7	1	1.10	214.7	1	1.10	214.7	CR+L
C206	214.7	1	1.75	214.7	1	1.75	214.7	CR+L
C207	214.5	1	2.70	214.5	1	2.70	214.5	CR+L
C208	214.5	1	1.85	214.5	1	1.85	214.5	CR+L
RC201	360.2	1	0.25	360.2	1	0.25	360.2	CR+L
RC202	338.0	1	0.58	338.0	1	0.58	338.0	CR+KLM
RC203	326.9	1	0.72	326.9	1	0.72	326.9	IV+C
RC204	299.7	1	1.95	299.7	1	1.95	299.7	C
RC205	338.0	1	0.62	338.0	1	0.62	338.0	L+KLM
RC206	324.0	1	0.87	324.0	1	0.87	324.0	KLM
RC207	298.3	1	0.88	298.3	1	0.88	298.3	KLM
RC208	269.1	1	78.42	269.1	1	78.42	269.1	C

Table 6: Instances with 25 customers.

	with ESPPRC			with ESPPRC and SR				
Instance	LB	Tree	Time (s)	LB	Tree	Time (s)	UB	Ref.
R101	1043.4	3	0.14	1044.0	1	0.09	1044.0	KDMSS
R102	909.0	1	0.27	909.0	1	0.27	909.0	KDMSS
R103	769.3	13	4.98	772.9	1	2.02	772.9	KDMSS
R104	619.1	21	33.29	625.4	1	6.73	625.4	KDMSS
R105	892.2	29	2.78	893.7	5	1.15	899.3	KDMSS
R106	791.4	5	1.41	793.0	1	0.83	793.0	KDMSS
R107	707.3	11	5.56	711.1	1	4.76	711.1	KDMSS
R108	594.7	789	1723.29	607.4	23	1601.68	617.7	CR+KLM
R109	775.4	77	20.11	783.3	7	11.54	786.8	KDMSS
R110	695.1	9	3.38	697.0	1	1.46	697.0	KDMSS
R111	696.3	41	19.21	707.2	1	3.67	707.2	CR+KLM
R112	614.9	165	169.26	630.2	1	35.67	630.2	CR+KLM
C101	362.4	1	0.47	362.4	1	0.47	362.4	KDMSS
C102	361.4	1	1.59	361.4	1	1.59	361.4	KDMSS
C103	361.4	1	6.06	361.4	1	6.06	361.4	KDMSS
C104	358.0	1	1564.88	358.0	1	1564.88	358.0	KDMSS
C105	362.4	1	0.49	362.4	1	0.49	362.4	KDMSS
C106	362.4	1	0.69	362.4	1	0.69	362.4	KDMSS
C107	362.4	1	0.97	362.4	1	0.97	362.4	KDMSS
C108	362.4	1	1.55	362.4	1	1.55	362.4	KDMSS
C109	362.4	1	3.62	362.4	1	3.62	362.4	KDMSS
RC101	850.1	39	5.60	944.0	1	2.12	944.0	KDMSS
RC102	721.9	127	60.41	822.5	1	8.68	822.5	KDMSS
RC103	645.3	9	8.56	710.9	1	40.05	710.9	KDMSS
RC104	545.8	1	5.71	545.8	1	5.71	545.8	KDMSS
RC105	761.6	21	7.22	855.3	1	4.31	855.3	KDMSS
RC106	664.5	11	3.35	723.2	1	3.88	723.2	KDMSS
RC107	603.6	7	4.60	642.7	1	4.49	642.7	KDMSS
RC108	541.2	5	15.88	594.8	5	260.95	598.1	KDMSS
R201	791.9	1	4.97	791.9	1	4.97	791.9	CR+KLM
R202	698.5	1	9.88	698.5	1	9.88	698.5	CR+KLM
R203	598.6	25	355.99	605.3	1	50.80	605.3	IV+C
R204			-			-	506.4	IV
R205	682.9	35	118.12	690.1	1	15.45	690.1	IV+C
R206	626.4	47	288.00	632.4	1	190.86	632.4	IV+C
R207	564.1	141	15400.44	575.5	1	34406.96	575.5	**JPSP**
R208			-			-	-	-
R209	599.9	3	24.45	600.6	1	16.63	600.6	IV+C
R210	636.1	49	332.70	645.3	3	18545.61	645.6	IV+C
R211	528.7	31	44644.89	535.5	1	10543.81	535.5	IV+DLP
C201	360.2	1	42.07	360.2	1	42.07	360.2	CR+L
C202	360.2	1	67.05	360.2	1	67.05	360.2	CR+KLM
C203	359.8	1	214.88	359.8	1	214.88	359.8	CR+KLM
C204			-			-	350.1	KLM
C205	359.8	1	64.18	359.8	1	64.18	359.8	CR+KLM
C206	359.8	1	38.91	359.8	1	38.91	359.8	CR+KLM
C207	359.6	1	72.81	359.6	1	72.81	359.6	CR+KLM
C208	350.5	1	55.79	350.5	1	55.79	350.5	CR+KLM
RC201	684.8	1	3.00	684.8	1	3.00	684.8	L+KLM
RC202	613.6	1	10.69	613.6	1	10.69	613.6	IV+C
RC203	555.3	1	190.88	555.3	1	190.88	555.3	IV+C
RC204			-			-	442.2	DLP
RC205	630.2	1	5.88	630.2	1	5.88	630.2	IV+C
RC206	610.0	1	8.17	610.0	1	8.17	610.0	IV+C
RC207	558.6	1	21.53	558.6	1	21.53	558.6	C
RC208			-	476.7	1	1639.40	476.7	S

Table 7: Instances with 50 customers.

	with ESPPRC			with ESPPRC and SR				
Instance	LB	Tree	Time (s)	LB	Tree	Time (s)	UB	Ref.
R101	1631.2	57	20.08	1634.0	3	1.87	1637.7	KDMSS
R102	1466.6	1	4.39	1466.6	1	4.39	1466.6	KDMSS
R103	1206.8	19	55.78	1208.7	1	23.85	1208.7	CR+L
R104			-	971.3	3	32343.92	971.5	IV
R105	1346.2	113	126.96	1355.2	5	43.12	1355.3	KDMSS
R106	1227.0	147	511.07	1234.6	1	75.42	1234.6	CR+KLM
R107			-	1064.3	3	1310.30	1064.6	CR+KLM
R108			-	932.1	1	5911.74	932.1	**JPSP**
R109			-	1144.1	19	1432.41	1146.9	CR+KLM
R110			-	1068.0	3	1068.31	1068.0	CR+KLM
R111			-	1045.9	39	83931.48	1048.7	CR+KLM
R112			-	946.7	9	202803.94	948.6	**JPSP**
C101	827.3	1	3.02	827.3	1	3.02	827.3	KDMSS
C102	827.3	1	12.92	827.3	1	12.92	827.3	KDMSS
C103	826.3	1	33.89	826.3	1	33.89	826.3	KDMSS
C104	822.9	1	4113.09	822.9	1	4113.09	822.9	KDMSS
C105	827.3	1	5.34	827.3	1	5.34	827.3	KDMSS
C106	827.3	1	7.15	827.3	1	7.15	827.3	KDMSS
C107	827.3	1	6.55	827.3	1	6.55	827.3	KDMSS
C108	827.3	1	14.46	827.3	1	14.46	827.3	KDMSS
C109	827.3	1	20.53	827.3	1	20.53	827.3	KDMSS
RC101	1584.1	59	56.62	1619.8	1	12.39	1619.8	KDMSS
RC102			-	1457.4	1	76.69	1457.4	CR+KLM
RC103			-	1257.7	3	2705.78	1258.0	CR+KLM
RC104			-	1129.9	7	65806.79	1132.3	IV
RC105	1472.0	191	309.83	1513.7	1	26.73	1513.7	KDMSS
RC106			-	1367.3	37	15891.55	1372.7	S
RC107			-	1207.8	1	153.80	1207.8	IV
RC108			-	1114.2	1	3365.00	1114.2	IV
R201			-	1143.2	1	139.03	1143.2	KLM
R202			-	1027.3	13	8282.38	1029.6	**JPSP**
R203			-	870.8	1	54187.40	870.8	**JPSP**
R204			-			-	-	-
R205			-			-	-	-
R206			-			-	-	-
R207			-			-	-	-
R208			-			-	-	-
R209			-	854.8	3	78560.47	854.8	**JPSP**
R210			-			-	-	-
R211			-			-	-	-
C201	589.1	1	203.34	589.1	1	203.34	589.1	CR+KLM
C202	589.1	1	3483.15	589.1	1	3483.15	589.1	CR+KLM
C203	588.7	1	13070.71	588.7	1	13070.71	588.7	KLM
C204			-			-	588.1	IV
C205	586.4	1	416.56	586.4	1	416.56	586.4	CR+KLM
C206	586.0	1	594.92	586.0	1	594.92	586.0	CR+KLM
C207	585.8	1	1240.97	585.8	1	1240.97	585.8	CR+KLM
C208	585.8	1	555.27	585.8	1	555.27	585.8	KLM
RC201			-	1261.7	3	229.27	1261.8	KLM
RC202			-	1092.3	1	312.57	1092.3	IV+C
RC203	922.6	11	34063.95	923.7	1	14917.36	923.7	**JPSP**
RC204			-			-	-	-
RC205			-	1154.0	1	221.24	1154.0	IV+C
RC206			-	1051.1	1	339.69	1051.1	**JPSP**
RC207			-			-	-	-
RC208			-			-	-	-

Table 8: Instances with 100 customers.

B Solutions of Closed Solomon Instances

Cost	Route
8.8	53
119.2	70, 30, 20, 66, 65, 71, 35, 34, 78, 77, 28
105.4	92, 98, 91, 44, 14, 38, 86, 16, 61, 85,100, 37
84.1	2, 57, 15, 43, 42, 87, 97, 95, 94, 13, 58
106.5	73, 22, 41, 23, 67, 39, 56, 75, 74, 72, 21, 40
114.6	52, 88, 62, 19, 11, 64, 63, 90, 32, 10, 31
78.4	6, 96, 59, 99, 93, 5, 84, 17, 45, 83, 60, 89
107.3	26, 12, 80, 68, 29, 24, 55, 4, 25, 54
93.2	27, 69, 76, 3, 79, 9, 51, 81, 33, 50, 1
114.6	18, 7, 82, 8, 46, 36, 49, 47, 48
932.1	10

Table 9: Solution of R108.100. The left column is the cost of the routes and the total cost. The right column is a comma separated list indicating the customers visited on the routes in the order of visit and the total number of routes.

Cost	Route
78.1	6, 94, 95, 87, 42, 43, 15, 57, 58
115.8	2, 41, 22, 75, 56, 23, 67, 39, 25, 55, 54
117.4	28, 76, 79, 78, 34, 35, 71, 65, 66, 20, 1
128.2	31, 62, 19, 11, 63, 64, 49, 36, 47, 48
62.8	53, 40, 21, 73, 74, 72, 4, 26
98.0	52, 88, 7, 82, 8, 46, 45, 17, 84, 5, 89
76.4	12, 80, 68, 24, 29, 3, 77, 50
100.5	61, 16, 86, 38, 14, 44, 91,100, 37, 59, 96
67.6	18, 83, 60, 99, 93, 85, 98, 92, 97, 13
103.8	27, 69, 33, 81, 9, 51, 30, 32, 90, 10, 70
948.6	10

Table 10: Solution of R112.100.

Cost	Route
8.8	53
93.6	52, 62, 63, 90, 10, 32, 70
177.2	83, 45, 82, 48, 47, 36, 19, 11, 64, 49, 46, 17, 5, 60, 89
223.8	50, 33, 65, 71, 29, 76, 3, 79, 78, 81, 9, 51, 20, 66, 35, 34, 68, 77
140.2	27, 69, 1, 30, 31, 88, 7, 18, 8, 84, 86, 91,100, 37, 98, 93, 59, 94
67.1	40, 73, 41, 22, 74, 2, 58
148.9	28, 26, 21, 72, 75, 39, 67, 23, 56, 4, 54, 55, 25, 24, 80, 12
170.0	95, 92, 42, 15, 14, 38, 44, 16, 61, 85, 99, 96, 6, 87, 57, 43, 97, 13
1029.6	8

Table 11: Solution of R202.100.

Cost	Route
24.2	53, 40, 58
142.1	27, 69, 1, 76, 3, 79, 78, 81, 9, 66, 71, 35, 34, 29, 68, 77, 28
187.3	89, 18, 45, 46, 36, 47, 48, 19, 11, 62, 88, 7, 82, 8, 83, 60, 5, 84, 17, 61, 91,100, 37, 98, 93, 59, 94
183.3	95, 92, 97, 42, 15, 43, 14, 44, 38, 86, 16, 85, 99, 96, 6, 87, 57, 41, 22, 74, 73, 2, 13
190.3	50, 33, 51, 71, 65, 20, 30, 32, 90, 63, 64, 49, 10, 70, 31, 52
143.6	26, 21, 72, 75, 39, 67, 23, 56, 4, 55, 25, 54, 24, 80, 12
870.8	6

Table 12: Solution of R203.100.

Cost	Route
202.5	27, 31, 7, 48, 47, 36, 46, 45, 8, 18, 6, 37, 44, 14, 38, 16, 17, 5, 13
130.5	2, 42, 43, 15, 23, 39, 22, 41, 21, 40
242.5	28, 12, 3, 33, 50, 1, 30, 11, 49, 19, 10, 32, 20, 9, 35, 34, 29, 24, 25, 4, 26
575.5	3

Table 13: Solution of R207.50.

Cost	Route
146.8	52, 7, 82, 83, 18, 6, 94, 13, 87, 57, 15, 43, 42, 97, 92, 37,100, 91, 93, 96
198.7	95, 99, 59, 98, 85, 5, 84, 61, 16, 44, 14, 38, 86, 17, 45, 8, 46, 36, 49, 48, 60, 89
205.9	27, 69, 31, 88, 62, 47, 19, 11, 64, 63, 90, 30, 51, 71, 9, 81, 33, 79, 3, 77, 68, 80, 24, 54, 26
157.6	28, 12, 76, 29, 78, 34, 35, 65, 66, 20, 32, 10, 70, 1, 50
145.8	40, 2, 73, 21, 72, 75, 23, 67, 39, 25, 55, 4, 56, 74, 22, 41, 58, 53
854.8	5

Table 14: Solution of R209.100.

Cost	Route
139.4	81, 54, 72, 37, 36, 39, 42, 44, 41, 38, 40, 35, 43, 61, 68
172.8	90, 65, 83, 64, 85, 63, 89, 76, 23, 21, 48, 18, 19, 49, 22, 20, 51, 84, 56, 66
241.4	69, 98, 88, 53, 82, 99, 52, 86, 87, 9, 10, 47, 17, 13, 74, 59, 97, 75, 58, 77, 25, 24, 57
211.0	1, 3, 5, 45, 60, 12, 11, 15, 16, 14, 78, 73, 79, 7, 6, 8, 46, 4, 2, 55,100, 70
159.1	91, 92, 95, 62, 33, 32, 30, 27, 26, 28, 29, 31, 34, 50, 67, 94, 93, 71, 96, 80
923.7	5

Table 15: Solution of RC203.100.

Cost	Route
8.4	90
186.6	81, 94, 67, 84, 85, 51, 76, 89, 48, 25, 77, 58, 74
168.6	92, 71, 72, 42, 39, 38, 36, 40, 44, 43, 41, 37, 35, 54, 93, 96
180.9	65, 83, 64, 95, 62, 63, 33, 30, 31, 29, 27, 28, 26, 32, 34, 50, 56, 91, 80
189.6	61, 2, 45, 5, 8, 7, 79, 73, 78, 53, 88, 6, 46, 4, 3, 1,100, 70, 68
120.9	82, 99, 52, 86, 57, 23, 21, 18, 19, 49, 20, 22, 24, 66
196.1	69, 98, 12, 14, 47, 16, 15, 11, 59, 75, 97, 87, 9, 13, 10, 17, 60, 55
1051.1	7

Table 16: Solution of RC206.100.

References

[1] C. Barnhart, E. L. Johnson, G. L. Nemhauser, M. W. P. Savelsbergh, and P. H. Vance. Branch-and-price: Column generation for solving huge integer programs. *Operations Research*, 46(3):316–329, 1998. doi: 10.1287/opre.46.3.316.

[2] J.E. Beasley and N. Christofides. An algorithm for the resource constrained shortest path problem. *Networks*, 19:379–394, 1989. doi: 10.1002/net.3230190402.

[3] N. Boland, J. Dethridge, and I. Dumitrescu. Accelerated label setting algorithms for the elementary resource constrained shortest path problem. *Operation Research Letters*, 34 (1):58–68, 2006. doi: 10.1016/j.orl.2004.11.011.

[4] A. Caprara, M. Fischetti, and A. N. Letchford. On the separation of maximally violated mod-k cuts. *Mathematical Programming*, 87(A):37–56, 1999. doi: 10.1007/s101079900107.

[5] A. Chabrier. Vehicle routing problem with elementary shortest path based column generation. *Computers & Operations Research*, 33(10):2972–2990, 2006. doi: 10.1016/j.cor.2005.02.029.

[6] V. Chvatal. Edmonds polytopes and hierarchy of combinatorial problems. *Discrete Mathematics*, 4(4):305–337, 1973. doi: 10.1016/0012-365X(73)90167-2.

[7] COIN. COIN — COmputational INfrastructure for Operations Research, 2005. http://www.coin-or.org.

[8] W. Cook and J. L. Rich. A parallel cutting plane algorithm for the vehicle routing problem with time windows. Technical Report TR99-04, Computational and Applied Mathematics, Rice University, Houston, Texas, USA, 1999.

[9] E. Danna and C. Le Pape. Branch-and-price heuristics: A case study on the vehicle routing problem with time windows. In G. Desaulniers, J. Desrosiers, and M. M. Solomon, editors, *Column Generation*, chapter 4, pages 99–129. Springer, 2005. doi: 10.1007/0-387-25486-2_4.

[10] G. Desaulniers, J. Desrosiers, J. Ioachim, I. M. Solomon, F. Soumis, and D. Villeneuve. A unified framework for deterministic time constrained vehicle routing and crew scheduling problems. In T. G. Crainic and G. Laporte, editors, *Fleet Management and Logistics*, pages 57–93. Kluwer, 1998.

[11] M. Desrochers. *La fabrication d2019horaires de travail pour les conducteurs d2019autobus par une mÃ©thode de gÃ©nÃ©ration de colonnes.* PhD thesis, UniversitÃ© de MontrÃ©al, MontrÃ©al, Canada, 1986.

[12] M. Desrochers, J. Desrosiers, and M. Solomon. A new optimization algorithm for the vehicle routing problem with time windows. *Operations Research*, 40(2):342–354, 1992. doi: 10.1287/opre.40.2.342.

[13] M. Dror. Note on the complexity of the shortest path models for column generation in VRPTW. *Operations Research*, 42(5):977–979, 1994. doi: 10.1287/opre.42.5.977.

[14] I. Dumitrescu. *Constrained Path and Cycle Problems*. PhD thesis, Department of Mathematics and Statistics, University of Melbourne, Australia, 2002.

[15] F. Eisenbrand. Note - on the membership problem for the elementary closure of a polyhedron. *Combinatorica*, 19(2):297–300, 1999. doi: 10.1007/s004930050057.

[16] D. Feillet, P. Dejax, M. Gendreau, and C. Gueguen. An exact algorithm for the elementary shortest path problem with resource constraints: Application to some vehicle routing problems. *Networks*, 44(3):216–229, 2004. doi: 10.1002/net.v44:3.

[17] R. Fukasawa, H. Longo, J. Lysgaard, M. Poggi de AragÃ£o, M. Reis, E. Uchoa, and R.F. Werneck. Robust branch-and-cut-and-price for the capacitated vehicle routing problem. *Mathematical Programming*, 106(3):491–511, 2006. doi: 10.1007/s10107-005-0644-x.

[18] S. Irnich. Resource extension functions: Properties, inversion, and generalization to segments. *OR Spectrum*, 30(1):113–148, 2008. doi: 10.1007/s00291-007-0083-6.

[19] S. Irnich and G. Desaulniers. Shortest path problems with resource constraints. In G. Desaulniers, Jacques Desrosiers, and M.M. Solomon, editors, *Column Generation*, chapter 2, pages 33–65. Springer, 2005. doi: 10.1007/0-387-25486-2_2.

[20] S. Irnich and D. Villeneuve. The shortest path problem with resource constraints and k-cycle elimination for $k \geq 3$. *INFORMS Journal on Computing*, 18:391–406, 2006. doi: 10.1287/ijoc.1040.0117.

[21] M. Jepsen, B. Petersen, and S. Spoorendonk. A branch-and-cut-and-price framework for the VRP applied on CVRP and VRPTW. Master's thesis, DIKU Department of Computer Science, University of Copenhagen, Denmark, 2005.

[22] B. Kallehauge, J. Larsen, and O.B.G. Madsen. Lagrangean duality and non-differentiable optimization applied on routing with time windows - experimental results. Technical Report Internal report IMM-REP-2000-8, Department of Mathematical Modelling, Technical University of Denmark, Lyngby, Denmark, 2000.

[23] N. Kohl, J. Desrosiers, O. B. G. Madsen, M. M. Solomon, and F. Soumis. 2-path cuts for the vehicle routing problem with time windows. *Transportation Science*, 33(1):101–116, 1999. doi: 10.1287/trsc.33.1.101.

[24] J. Larsen. *Parallelization of the vehicle routing problem with time windows*. PhD thesis, Department of Mathematical Modelling, Technical University of Denmark, Lyngby, Denmark, 1999.

[25] J. Lysgaard. CVRPSEP: A package of separation routines for the capacitated vehicle routing problem. Working Paper 03-04, Dept. of Mgt. Science and Logistics, Aarhus School of Business., 2003.

[26] J. Lysgaard, A. N. Letchford, and R. W. Eglese. A new branch-and-cut algorithm for the capacitated vehicle problem. *Mathematical Programming*, 100(2):423–445, 2004. doi: 10.1007/s10107-003-0481-8.

[27] M. E. LÃ$\frac{1}{4}$bbecke and J. Desrosiers. Selected topics in column generation. *Operations Research*, 53(6):1007–1023, 2005. doi: 10.1287/opre.1050.0234.

[28] G. Nemhauser and S. Park. A polyhedral approach to edge coloring. *Operations Research Letters*, 10(6):315–322, 1991. doi: 10.1016/0167-6377(91)90003-8.

[29] G. L. Nemhauser and L. A. Wolsey. *Integer and Combinatorial Optimization*. John Wiley & Sons, Inc., 1988.

[30] D. Pisinger and L. B. Reinhardt. Multi-objective non-additive shortest path. DIKU Department of Computer Science, University of Copenhagen, Denmark, submitted, 2007.

[31] G. Righini and M. Salani. Symmetry helps: bounded bi-directional dynamic programming for the elementary shortest path problem with resource constraints. *Discrete Optimization*, 3(3):255–273, 2006. doi: 10.1016/j.disopt.2006.05.007.

[32] G. Righini and M. Salani. New dynamic programming algorithms for the resource constrained shortest path problem. *Networks*, 51(3):155–170., 2008. doi: 10.1002/net.20212.

[33] M. Salani. *Branch-and-Price Algorithms for Vehicle Routing Problems*. PhD thesis, Universitá Degli Studi Di Milano, Facoltá di Scienza Matematiche, Fisuche e Naturali Dipartimento di Technologie dell'Informazione, Milano, Italy, 2005.

[34] M. M. Solomon. Algorithms for the vehicle routing and scheduling problem with time window constraints. *Operations Research*, 35(2):234–265, 1987. doi: 10.1287/opre.35.2.254.

[35] SPEC. Standard Performance Evaluation Corporation, 2005. http://www.spec.org.

[36] P. Toth and D. Vigo, editors. *The Vehicle Routing Problem*. SIAM, 2002.

[37] L. A. Wolsey. *Integer Programming*. John Wiley & Sons, Inc., 1998.

Chapter 4

Chvátal-Gomory Rank-1 Cuts used in a Dantzig-Wolfe Decomposition of the Vehicle Routing Problem with Time Windows

Bjørn Petersen
DIKU Department of Computer Science, University of Copenhagen

David Pisinger
DIKU Department of Computer Science, University of Copenhagen

Simon Spoorendonk
DIKU Department of Computer Science, University of Copenhagen

Abstract

This chapter shows how Chvátal-Gomory (CG) rank-1 cuts can be used in a branch-and-cut-and-price algorithm for the vehicle routing problem with time windows (VRPTW). Using Dantzig-Wolfe decomposition we split the problem into a set partitioning problem as master problem and an elementary shortest path problem with resource constraints as pricing problem. To strengthen the formulation we derive general CG rank-1 cuts based on the master problem formulation. Adding these cuts to the master problem means that an additional resource is added to the pricing problem for each cut. This increases the complexity of the label algorithm used to solve the pricing problem since normal dominance tests become weak when many resources are present and hence most labels are incomparable. To overcome this problem we present a number of improved dominance tests exploiting the step-like structure of the objective function of the pricing problem. Computational experiments are reported on the Solomon test instances showing that the addition of CG rank-1 cuts improves the lower bounds significantly and makes it possible to solve a majority of the instances in the root node of the branch-and-bound tree. This indicates that CG rank-1 cuts may be essential for solving future large-scale VRPTW

Published in the book *The Vehicle Routing Problem: Latest Advances and New Challenges*, 2008

problems where we cannot expect that the branching process will close the gap between lower and upper bounds in reasonable time.

Keywords: Vehicle routing problem with time windows, Dantzig-Wolfe decomposition, Chvatal-Gomory rank-1 cuts.

1 Introduction

In the vehicle routing problem with time windows (VRPTW) we are given a set of customers with an associated demand and a number of identical vehicles. The task is to find a set of minimum-length routes starting and ending at a central depot such that each customer is visited exactly once within a given time window, and the capacity of each vehicle is respected.

The standard Dantzig-Wolfe decomposition of the arc flow formulation of the VRPTW is to split the problem into a master problem (a set partitioning problem with a convexity constraint, stating that all customers should be visited with a limited number of vehicles) and a pricing problem (an elementary shortest path problem with resource constraints (ESPPRC), where capacity and time are the constrained resources). Delayed column generation may be used to solve the LP-relaxed master problem, which can be used as lower bound in a branch-and-bound algorithm to reach integrality. Applying cutting planes either in the master or the pricing problem leads to a branch-and-cut-and-price algorithm (BCP).

BCP algorithms have been frequently used to solve the VRPTW, e.g., Kohl et al. [25], Cook and Rich [6], Larsen [26], Kallehauge et al. [24], Irnich and Villeneuve [22], Chabrier [4], Danna and Le Pape [7], Salani [31]. In all cases the valid inequalities have been based on the *original* arc flow formulation of the VRPTW, i.e., the inequalities added are valid for both the *original* arc formulation and the master problem. Fukasawa et al. [16] refer to this as a *robust* approach. Recently Jepsen et al. [23] showed how the subset row (SR) inequalities, which are valid inequalities for the set partitioning problem, successfully can be applied to VRPTW in a column generation context. In their computational results they report solving 8 out of 18 previously unsolved instances from the set of benchmarks by Solomon [33]. In a following paper Desaulniers et al. [9] added fast pricing heuristics and improved cutting policies for the SR inequalities to obtain even better results by closing an additional 5 instances. The latter approaches are denoted *non-robust* according to the classification by Fukasawa et al. [16], since the complexity of the pricing problem is increased when SR inequalities are added to the master problem.

Jepsen et al. [23] showed that the separation of SR inequalities is $\mathcal{NP}$-hard and that the inequalities can be recognized as a subset of the Chvátal-Gomory (CG) rank-1 cuts. A simple enumeration algorithm was used to separate the SR inequalities for sets of rows of size three, and even for such small sets the computational results were very good as mentioned above. Not surprisingly the separation of CG rank-1 cuts is also known to be $\mathcal{NP}$-hard, see Eisenbrand [13]. Fischetti and Lodi [15] used the CG rank-1 cuts as cutting planes in an integer problem and showed how the separation can be formulated as a mixed integer problem. They obtained lower bounds when optimizing over the first Chvátal closure, i.e., adding violated CG rank-1 cuts, and were the first to report an optimal solution to one instance from MIPLIB 3.0 by Bixby et al. [1]. These results motivate the incorporation of the CG rank-1 cuts in a BCP algorithm.

The pricing problem of the Dantzig-Wolfe decomposition of VRPTW, i.e., the ESPPRC, was shown to be $\mathcal{NP}$-hard by Dror [11]. Commonly the ESPPRC has been solved with labeling

algorithms, see Dumitrescu [12], Feillet et al. [14], Righini and Salani [29, 30], Boland et al. [2]. Due to the difficulty of the ESPPRC most earlier approaches solved relaxations of the ESPPRC, see Desrochers et al. [10], Irnich and Villeneuve [22]. For a general introduction to resource constrained shortest path problems, see Desaulniers et al. [8], Irnich and Desaulniers [21], Irnich [20]. Jepsen et al. [23] provides an introduction of the SR inequalities and how their application in the master problem leads to an additional resource per inequality in the pricing problem. Furthermore, it is shown how the dominance criteria of the label algorithm can be improved.

In this chapter we extend the work by Jepsen et al. [23] to include general CG rank-1 cuts for the Set Partitioning master problem. Each cut results in a new resource constraint in the ESPPRC pricing problem. As the resource extension functions are non-decreasing any dynamic programming algorithm for the ESPPRC can be used to solve the resulting problem. However, the addition of new resources means that more labels become incomparable when using a traditional dominance test, and hence the number of labels in the dynamic programming explodes. To overcome this problem we exploit the fact that in the pricing problem it is sufficient to find a cost-minimal solution, and not all Pareto-optimal solutions. Due to this fact we may temporarily replace each label with a number of equivalent labels such that resources become comparable in the dominance test. This approach considerably decreases the number of labels generated in the dynamic programming algorithm. As demonstrated in the computational results we can in this way solve the ESPPRC pricing problem even when several hundreds of CG rank-1 cuts have been added, and hence several hundreds of resources are to be dealt with in the label algorithm.

The chapter is organized as follows: In Section 2 we give an overview of the Dantzig-Wolfe decomposition of the VRPTW and describe how to calculate the reduced cost of columns when delayed column generation is used. For completeness we review the CG rank-1 cuts and their separation, as described by Fischetti and Lodi [15], in Section 3. Furthermore, we clarify how to use these techniques in a VRPTW context. In Section 4 the improved dominance criteria of the label algorithm are described. An algorithmic outline, implementation details, and computational results using the Solomon benchmark instances are presented in Section 5. Section 6 provides some concluding remarks.

2 Decomposition

Let C be the set of customers, and let the set of nodes be $V = C \cup \{o, o'\}$ where o denotes the depot at the start of the routes and o' denotes the depot at the end. Each customer $i \in C$ has a demand d_i while we set $d_o = d_{o'} = 0$. Each node $i \in V$ has an associated service s_i and a time windows $[a_i, b_i]$ in which it should be visited.

Let $E = \{(i, j) : i, j \in V,\ i \neq j\}$ be the set of arcs between the nodes. The set of vehicles K is sufficiently large, e.g., $|K| = V$, such that the convexity constraint is not binding, and each vehicle has capacity D. If vehicle $k \in K$ service node $i \in V$ then the variable t_{ik} denotes the arrival time of the vehicle. Let c_{ij} be the travel cost on arc $(i, j) \in E$ and let x_{ijk} be the variable indicating whether vehicle $k \in K$ traverses arc $(i, j) \in E$. The overall travel time τ_{ij} on arc $(i, j) \in E$ depends on the travel time of the arc and the service time s_i at customer i.

The 3-index flow model (Toth and Vigo [34]) for the VRPTW becomes:

$$\min \sum_{k \in K} \sum_{(i,j) \in E} c_{ij} x_{ijk} \tag{1}$$

$$\text{s.t.} \sum_{k \in K} \sum_{(i,j) \in \delta^+(i)} x_{ijk} = 1 \qquad \forall i \in C \tag{2}$$

$$\sum_{(i,j) \in \delta^+(o)} x_{ijk} = \sum_{(i,j) \in \delta^-(o')} x_{ijk} = 1 \qquad \forall k \in K \tag{3}$$

$$\sum_{(j,i) \in \delta^-(i)} x_{jik} - \sum_{(i,j) \in \delta^+(i)} x_{ijk} = 0 \qquad \forall i \in C,\ \forall k \in K \tag{4}$$

$$\sum_{(i,j) \in E} d_i x_{ijk} \leq D \qquad k \in K \tag{5}$$

$$a_i \leq t_{ik} \leq b_i \qquad \forall i \in V,\ \forall k \in K \tag{6}$$

$$x_{ijk}(t_{ik} + \tau_{ij}) \leq t_{jk} \qquad \forall (i,j) \in E,\ \forall k \in K \tag{7}$$

$$x_{ijk} \in \{0,1\} \qquad \forall (i,j) \in E,\ \forall k \in K \tag{8}$$

Constraints (2) ensure that every customer $i \in C$ is visited, and (3) ensures that each route starts and ends in the depot. Constraint set (4) ensure flow conservation for each vehicle k. Note that a zero-cost arc $x_{oo'k}$ between the start and end depot must be present for all vehicles to allow an empty tour in case not all vehicles are needed. The constraint set (5) ensures that the capacity of each vehicle is not exceeded and constraint sets (6) and (7) ensure that the time window constraints are satisfied. Note that (7) together with the assumption that $\tau_{ij} > 0$ for all $(i,j) \in E$ eliminates all sub-tours. The last constraint define the domain of the arc flow variables.

The standard Dantzig-Wolfe decomposition of the VRPTW, see e.g. Desrochers et al. [10], leads to the following master problem:

$$\min \sum_{p \in P} \sum_{(i,j) \in E} c_{ij} \alpha_{ijp} \lambda_p \tag{9}$$

$$\text{s.t} \sum_{p \in P} \sum_{(i,j) \in \delta^+(i)} \alpha_{ijp} \lambda_p = 1 \qquad \forall i \in C \tag{10}$$

$$\lambda_p \in \{0,1\} \qquad \forall p \in P \tag{11}$$

where P is the set of all feasible routes, the binary constant α_{ijp} is one if and only if arc (i,j) is used by route $p \in P$, and the binary variable λ_p indicates whether route p is used. The master problem is a set partitioning problem and the LP relaxation can be solved using delayed column generation, i.e., consider a *restricted* master problem containing a subset of the columns P and generate additional columns as needed. For the remainder of this chapter the master problem will refer to the the restricted problem. Let $\pi_i \in \mathbb{R}$ for all $i \in C$ be the dual values of (10) and let $\pi_0 = 0$. Then the reduced cost of a route p is:

$$\overline{c}_p = \sum_{(i,j) \in E} c_{ij} \alpha_{ijp} - \sum_{(i,j) \in E} \pi_j \alpha_{ijp} = \sum_{(i,j) \in E} (c_{ij} - \pi_j) \alpha_{ijp} \tag{12}$$

The pricing problem is an ESPPRC where the cost of each arc is $\overline{c}_{ij} = c_{ij} - \pi_j$ for all arcs $(i,j) \in E$.

Valid inequalities based on the VRPTW constraints (2)-(8), i.e.,

$$\sum_{k \in K} \sum_{(i,j) \in E} \beta_{ij} x_{ijk} \leq \beta_0 \tag{13}$$

are handled as follows (Note that β_{ij} can be dependent on a vehicle k but then different pricing problems must be considered). Let μ be the dual values of (13), then an additional $\mu\beta_{ij}$ for all arcs $(i,j) \in E$ has to be subtracted from the reduced cost of a route, i.e., by subtracting the dual value from the arc cost in the the pricing problem, i.e., $\overline{c}_{ij} = c_{ij} - \pi_j - \mu\beta_{ij}$.

Consider adding a valid inequality for the set partitioning master problem (10)–(11) that cannot be written as a linear combination of the arc flow variables, i.e.,

$$\sum_{p \in P} \beta_p \lambda_p \leq \beta_0 \tag{14}$$

Let $\sigma \leq 0$ be the dual values of (14), then an additional $\sigma\beta_p$ has to be subtracted when calculating the reduced cost of the column, i.e, the new reduced cost is $\hat{c}_p = \overline{c}_p - \sigma\beta_p$. To handle the cost $-\sigma\beta_p$ it is necessary to modify the pricing problem by adding constraints or variables, thereby increasing its complexity.

3 Chvátal-Gomory Rank-1 Cuts

CG cuts are well known valid inequalities for integer programming problems, see Gomory [17], Chvatal [5]. However, in a BCP context these cuts have been given little attention. Except for the recent papers by Jepsen et al. [23], Desaulniers et al. [9] only an early attempt by Nemhauser and Park [28] has been found where general mixed-integer cuts for the master problem is applied. Nemhauser and Park [28] solved the pricing problem as a MIP by adding additional variables and constraints to take the dual values of the applied cuts into account. As noted in Jepsen et al. [23], the SR inequalities are a subset of the CG cuts, and since the SR inequalities were successfully used for VRPTW an obvious extension is to include a larger set of the CG cuts into the BCP framework. Hence, in the following the focus will be on the CG rank-1 cuts and their separation starting with the general case as described by Fischetti and Lodi [15]. Next we specify the form of CG rank-1 cuts for the master problem of the VRPTW and formulate the separation problem based the presented theory. Last we briefly discuss the interpretation of the SR inequalities with regards to the CG cuts.

Consider an IP problem:

$$\min\{c\lambda : A\lambda \leq b, \lambda \geq 0, \lambda \in \mathbb{Z}^n\}$$

where A is a $m \times n$ matrix, $N = 1, \ldots, n$ is the set of indices of variables, and $M = 1, \ldots, m$ is the set of indices of constraints. The two polyhedra

$$P_{LP} = \{\lambda \in \mathbb{R}^n : A\lambda \leq b, \lambda \geq 0\}$$
$$P_{IP} = conv\{\lambda \in \mathbb{Z}^n : A\lambda \leq b, \lambda \geq 0\} = conv(P_{LP} \cap \mathbb{Z}^n)$$

describe the solution space of the linear relaxation P_{LP} and the convex hull of the integer solutions in P_{LP}. It is assumed that all coefficients of A and b are integer. A CG cut is a valid inequality for P_{IP} given as:

$$\lfloor uA \rfloor \lambda \leq \lfloor ub \rfloor$$

where $u \geq 0$ is called the CG multiplier vector. The inequality is said to have *rank-1* with respect to $A\lambda \leq b$ and $\lambda \geq 0$. Higher rank cuts are obtained by considering systems that also contain lower rank CG cuts, e.g., a rank-2 cut is based on $A\lambda \leq b$ and $\lambda \geq 0$ and some rank-1 cuts. Note that given the above assumptions on the integrality of A and b, undominated CG cuts only arise for rational CG multipliers $u_i \in [0, 1)$, for all $i = 1, \ldots, m$, see Schrijver [32].

The first *Chvátal closure* of P_{LP} is defined as the polyhedron:

$$P_1 = \{\lambda \leq 0 : A\lambda \leq b, \lfloor uA \rfloor \lambda \leq \lfloor ub \rfloor, u \geq 0 \; \forall u \in \mathbb{R}^n\}$$

Clearly $P_{IP} \subseteq P_1 \subseteq P_{LP}$ but even more interesting is it, that $P_1 \subset P_{LP}$ iff $P_{IP} \neq P_{LP}$. The better approximation of P_{IP} is obtained, since it is possible to use a CG cut to cut off a fractional vertex $\lambda^* \in P_{LP}$ corresponding to the basis B by choosing multipliers u equal to the ith row of B^{-1} where i is the row associated with any fractional part of λ^*, see Gomory [17, 18].

The separation problem is stated by Fischetti and Lodi [15] as:

Definition 1. *Given a point $\lambda* \in P_{LP}$. The CG separation problem consists of finding a CG cut that is violated by λ^*, i.e., find $u \geq 0$ for $u \in \mathbb{R}^n$ such that $\lfloor uA \rfloor \lambda > \lfloor ub \rfloor$, or prove that no such u exist.*

Eisenbrand [13] showed that the separation problem is $\mathcal{NP}$-hard and computational results performed by Fischetti and Lodi [15] indicate that separation times can be cumbersome.

Given a fractional solution $\lambda^* \in P_{LP}$ the maximally violated CG cut $\gamma\lambda \leq \gamma_0$, where $\gamma = \lfloor uA \rfloor$ and $\gamma_0 = \lfloor ub \rfloor$ for some CG multipliers $u \geq 0$ for $u \in \mathbb{R}^n$ can be found by solving the following MIP:

$$\max \gamma\lambda^* - \gamma_0 \tag{15}$$
$$\gamma_j \leq uA_j \qquad \forall j \in N \tag{16}$$
$$\gamma_0 > ub - 1 \tag{17}$$
$$u_i \geq 0 \qquad \forall i \in M \tag{18}$$
$$\gamma_j \in \mathbb{Z} \qquad \forall j \in N \cup \{0\} \tag{19}$$

Note that only basis variables with non-zero values can contribute to the violation of the CG rank-1 cut. Hence, all zero valued variables can be left out of the formulation and their coefficients can be calculated after the CG multipliers are identified. This reduces the size of the MIP problem in both the number of variables and constraints.

Furthermore Fischetti and Lodi [15] suggest to reformulate the problem in order to obtain a stronger formulation and numerical stability. Based on the fact that the CG multipliers of undominated cuts are less than 1, bounding them from above provides a stronger formulation. However, later observations showed that the MIP heuristics performed much better without these bounds. To obtain numerical stability a slack variable $f_j \in [0, 1 - \delta]$ (e.g., $\delta = 0.01$) is introduced for each coefficient α_j.

Equivalent solutions to the separation problem can result in CG rank-1 cuts of different strength with respect to P_{IP}. A strong cut tends to be sparse, i.e., the number of non-zero entries is small. In order to obtain stronger and sparser cuts the objective function is modified by adding a small penalty w_i (e.g., $w_i = 0.0001$) for the selection of a multiplier u_i.

Let $N(\lambda^*)$ is the set of non-zero basis variables. This leads to the following formulation of the separation problem:

$$\max \sum_{j \in N(\lambda^*)} \gamma_j \lambda_j^* - \gamma_0 - \sum_{i \in M} w_i u_i \tag{20}$$

$$f_j = uA_j - \gamma_j \qquad \forall j \in N(\lambda^*) \tag{21}$$

$$f_0 = ub - \gamma_0 \tag{22}$$

$$0 \leq f_j \leq 1 - \delta \qquad \forall j \in N(\lambda^*) \cup \{0\} \tag{23}$$

$$u_i \geq 0 \qquad \forall i \in M \tag{24}$$

$$\gamma_j \in \mathbb{Z} \qquad \forall j \in N(\lambda^*) \cup \{0\} \tag{25}$$

The model (20)-(25) can be modified to handle systems as $A\lambda \geq b$ and $A\lambda = b$ by modifying the bounds of the CG multipliers, i.e., removing (24) and letting u be a free variables is a way to handle equations.

For VRPTW the the CG rank-1 cuts are based on the master problem constraints (10). The set partitioning constraints give rise to cuts with CG multipliers $u \in \mathbb{R}^{|C|}$, since they are equalities. However, since the CG cuts will be used in a column generation context two equally sparse cuts at separation time might not be equally sparse after column generation. This is especially the case for CG rank-1 cuts with negative multipliers in a minimization problem, where cuts tend to become very dense when columns price into the master problem. Hence, we restrict ourselves to consider CG rank-1 cuts with non-negative multipliers for the VRPTW.

The CG rank-1 cuts for the VRPTW with respect to the master problem (9)-(11) and with non-negative CG multipliers are given as:

$$\sum_{p \in P} \left\lfloor \sum_{i \in C} u_i \sum_{(i,j) \in \delta^+(i)} \alpha_{ijp} \right\rfloor \lambda_p \leq \left\lfloor \sum_{i \in C} u_i \right\rfloor \tag{26}$$

Given a fractional solution λ^* for the master problem (9)-(11) the most violated CG cut of rank-1 can be found by solving the following MIP:

$$\max \sum_{p \in P(\lambda^*)} \gamma_p \lambda_p^* - \gamma_0 - \sum_{i \in C} w_i u_i \tag{27}$$

$$f_p = \sum_{(i,j) \in \delta(i)^+} \alpha_{ijp} u_i - \gamma_p \qquad \forall p \in P(\lambda^*) \tag{28}$$

$$f_0 = \sum_{i \in C} u_i - \gamma_0 \tag{29}$$

$$0 \leq f_p \leq 1 - \delta \qquad \forall p \in P(\lambda^*) \cup \{0\} \tag{30}$$

$$0 \leq u_i \qquad \forall i \in C \tag{31}$$

$$\gamma_j \in \mathbb{Z}^+ \qquad \forall p \in P(\lambda^*) \cup \{0\} \tag{32}$$

Again it is possible to reduce the number of variables by only considering the non-zero basis variables.

From Jepsen et al. [23] we recall the SR inequalities for the VRPTW based on the master problem (9)-(11):

$$\sum_{p \in P} \left\lfloor \frac{1}{k} \sum_{i \in S} \sum_{(i,j) \in \delta^+(i)} \alpha_{ijp} \right\rfloor \lambda_p \leq \left\lfloor \frac{1}{k} |S| \right\rfloor \tag{33}$$

where $S \subseteq C$ and $0 < k \leq |S|$. This is equivalent to the set of CG rank-1 cuts where $|S|$ of the CG multipliers are equal to $\frac{1}{k}$ and the rest are equal to 0, i.e., a very sparse CG multiplier vector. A SR cut can also be interpreted as a mod-k cut proposed by Caprara et al. [3]. The mod-k cuts are CG rank-1 cuts with multipliers in the set $\{0, \frac{1}{k}, \ldots, \frac{k-1}{k}\}$, i.e., a SR cut is a mod-k cut with $|S|$ multipliers equal to $\frac{1}{k}$ and the rest are equal to 0. Extending the SR cut to allow a row (customer) to be present multiple times in S, i.e., let S be a multiset, leads to an SR cut with maximal $|S|$ multipliers in the set $\{0, \frac{1}{k}, \ldots, \frac{k-1}{k}\}$. That is, the CG multiplier of a row is raised by $\frac{1}{k}$ for each time it is present in S. This is indeed also a mod-k cut.

4 Label Algorithm

Finding a route with negative reduced cost in the pricing problem corresponds to finding a negative reduced cost path starting and ending at the depot, i.e., an ESPPRC. In the following sections we formally describe the ESPPRC and show how the pricing problem can be solved when new resources are introduced as a consequence of adding CG cuts.

4.1 The Pricing Problem

Assuming that no cuts have been added, the ESPPRC can be formally defined as: Given a weighted directed graph $G(V, E)$ with nodes V and arcs E, and a set of resources R. For each arc $(i, j) \in E$ and resource $r \in R$ three parameters are given: A lower limit $a_r(i, j)$ on the accumulation of resource r when traversing arc $(i, j) \in E$; an upper limit $b_r(i, j)$ on the accumulation of resource r when traversing arc $(i, j) \in E$; and finally an amount $c_r(i, j)$ of resource r consumed by traversing arc $(i, j) \in E$. The objective is to find a minimum cost path p from a source node $o \in V$ to a target node $o' \in V$, where the accumulated resources of p satisfy the limits for all resources $r \in R$. Without loss of generality we assume that the limits must be satisfied at the end of each arc (i, j), i.e., after $c_r(i, j)$ has been consumed.

If the nodes have associated some resource consumptions and some upper and lower limits on the accumulated resources are present, these can be expressed by equivalent resource constraints on the arcs (e.g. the incoming arcs of the node).

For the pricing problem of VRPTW the resources are load d, time t, and a binary visit-counter for each customer $v \in C$. When considering the pricing problem of VRPTW, the consumptions and upper and lower limits of the resources at each arc (i, j) in ESPPRC are:

$$\begin{array}{llll}
a_d(i,j) = 0, & b_d(i,j) = D - d_j, & c_d(i,j) = d_j & \forall (i,j) \in E \\
a_t(i,j) = a_i, & b_t(i,j) = b_i, & c_t(i,j) = \tau_{ij} & \forall (i,j) \in E \\
a_v(i,j) = 0, & b_v(i,j) = 1, & c_v(i,j) = 1 & \forall v \in V : v = j, \ \forall (i,j) \in E \\
a_v(i,j) = 0, & b_v(i,j) = 1, & c_v(i,j) = 0 & \forall v \in V : v \neq j, \ \forall (i,j) \in E
\end{array}$$

In the label algorithm labels at node v represent partial paths from o to v. The following attributes for a label L are considered:

$\overline{v}(L)$ The current end-node of the partial path represented by L.
$\overline{c}(L)$ The sum of the reduced cost along path L.
$r(L)$ The accumulated consumption of resource $r \in R$ along path L.

A feasible extension $\epsilon \in \mathcal{E}(L)$ of a label L is a partial path starting in node $\overline{v}(L) \in V$ and ending in the target node o' without violating any resource constraints when concatenated with the partial path represented by L.

In the following it is assumed that all resources are bounded strongly from above, and weakly from below. This means that if the current resource accumulation of a label is below the lower limit on a given arc, it is allowed to fill up the resource to the lower limit, i.e., waiting for a time window to open. This means that two consecutive labels L_u and L_v related by an arc (u, v), i.e., L_u is extended and creates L_v, where $\overline{v}(L_u) = u$ and $\overline{v}(L_v) = v$, must satisfy

$$r(L_v) \leq b_r(u, v), \qquad \forall r \in R \tag{34}$$
$$r(L_v) = \max\{r(L_u) + c_r(u, v), a_r(u, v)\}, \qquad \forall r \in R \tag{35}$$

Here (34) demands that each label L_v satisfies the upper limit $b_r(u, v)$ of resource r corresponding to arc (u, v), while (35) states that resource r of L_v corresponds to the resource consumption at label L_u plus the amount consumed by traversing arc (u, v), respecting the lower limit $a_r(u, v)$ on arc (u, v). Other authors refer to (35) as a *resource extension function*, see e.g. Desaulniers et al. [8].

A simple enumeration algorithm could be used to produce all these labels, but to limit the number of labels considered, dominance rules are introduced to fathom labels which do not lead to an optimal solution.

Definition 2. *A label* L_i dominates *label* L_j *if*

$$\overline{v}(L_i) = \overline{v}(L_j) \tag{36}$$
$$\overline{c}(L_i) \leq \overline{c}(L_j) \tag{37}$$
$$\mathcal{E}(L_j) \subseteq \mathcal{E}(L_i) \tag{38}$$

In other words, the paths corresponding to labels L_i and L_j should end at the same node $\overline{v}(L_i) = \overline{v}(L_j) \in V$, the path corresponding to label L_i should cost no more than the path corresponding to label L_j, and finally any feasible extension of L_j is also a feasible extension of L_i. Notice that we are only interested in one cost-minimal path and not all pareto-optimal paths, hence our dominance rule is tighter than the one used in e.g. Desaulniers et al. [8], Irnich and Desaulniers [21].

Feillet et al. [14] suggested to consider the set of nodes that cannot be reached from a label L_i and compare the set with the unreachable nodes of a label L_j in order to determine if some extensions are impossible and thereby potentially dominate where else not possible, since $v_{old}(L_i) \leq v_{old}(L_j) \Rightarrow v_{new}(L_i) \leq v_{new}(L_j)$ but $v_{new}(L_i) \leq v_{new}(L_j) \not\Rightarrow v_{old}(L_i) \leq v_{old}(L_j)$. Or in other words: update the node resources in an eager fashion instead of a lazy one. The following definition is a generalization of Feillet et al. [14][Definition 3].

Definition 3. *Given a start node* $o \in V$*, a label* L*, and a node* $u \in V$ *where* $\overline{v}(L) = u$ *a node* $v \in V$ *is considered* unreachable *if* v *has already been visited on the path from* o *to* u *or if a resource window is violated, i.e.:*

$$\exists r \in R \qquad\qquad r(L) + \ell_r(u, v) > b_r(v)$$

where $\ell_r(u,v)$ is a lower bound on the consumption of resource r on all feasible paths from u to v. The node resources *are then given as: $v(L) = 1$ indicates that node $v \in V$ is unreachable from node $\overline{v}(L) \in V$, and $v(L) = 0$ otherwise.*

To determine if (38) holds can be quite cumbersome, as the straightforward definition demands that we calculate all extensions of the two labels. Therefore a sufficient criterion for (38) to hold is sought which can be computed faster. If label L_i has consumed less resources than label L_j then no resources are limiting the possibilities of extending L_i compared to L_j, hence the following proposition can be used as a relaxed version of the dominance criteria.

Proposition 4. Desaulniers et al. [8]. *If all resource extension functions are non-decreasing, then label L_i dominates label L_j if:*

$$\overline{v}(L_i) = \overline{v}(L_j) \tag{39}$$

$$\overline{c}(L_i) \leq \overline{c}(L_j) \tag{40}$$

$$r(L_i) \leq r(L_j) \qquad \forall r \in R \tag{41}$$

Using Proposition 4 as a dominance criteria is a relaxation of the dominance criteria of Definition 2 since only a subset of labels satisfying (36), (37), and (38) satisfies inequalities (39), (40), and (41).

4.2 Solving the Pricing Problem with New Resources

Recall that a CG rank-1 cut (26) for the VRPTW master problem (9)–(11) is:

$$\sum_{p \in P} \left\lfloor \sum_{i \in C} u_i \sum_{(i,j) \in \delta^+(i)} \alpha_{ijp} \right\rfloor \lambda_p \leq \left\lfloor \sum_{i \in C} u_i \right\rfloor$$

Let $\sigma \leq 0$ be the corresponding dual variable when solving the master problem to LP-optimality. The reduced cost of column p in the VRPTW master problem is:

$$\hat{c}_p = \overline{c}_p - \sigma \left\lfloor \sum_{i \in C} u_i \sum_{(i,j) \in \delta^+(i)} \alpha_{ijp} \right\rfloor = \sum_{(i,j) \in E} \overline{c}_{ij} \alpha_{ijp} - \sigma \left\lfloor \sum_{i \in C} u_i \sum_{(i,j) \in \delta^+(i)} \alpha_{ijp} \right\rfloor$$

We analyze how this additional cost can be handled in the label algorithm for ESPPRC.

Let $V(L) = \{i \in V : i(L) = 1\}$ be the nodes visited on the partial path of label L. The reduced cost of L can then be expressed as:

$$\hat{c}(L) = \overline{c}(L) - \sigma \left\lfloor \sum_{i \in V(L)} u_i \right\rfloor \tag{42}$$

A new resource m can be used to compute the coefficient of penalty σ for label L, i.e., $m(L) = \sum_{i \in V(L)} u_i$, is the unfloored amount involved in the cut. Note that the consumption of resource m is u_i for each outgoing (incoming) arc of the customers $i \in C$. Even though the update of resource $\hat{c}$ is defined by a decreasing function, the usual dominance criteria of Proposition 4 can still be used, because in case L_i dominates L_j, $\overline{c}(L_i) \leq \overline{c}(L_j)$ and $m(L_i) \leq m(L_j)$ so $\hat{c}(L_i) \leq \hat{c}(L_j)$ since $-\sigma > 0$. Note that the resource $\hat{c}$ can be ignored

during the label algorithm and only be considered at the last arc to the target node to compute the reduced cost $\hat{c}(L)$ of path L from $\overline{c}(L)$ and $m(L)$.

Since all resource extension functions (including $m(L)$) are non-decreasing we can apply the label algorithm described in the previous section to solve the ESPPRC, using the dominance rule from Proposition 4 for the extended set of resources. However, as further cuts are added and hence more resources are to be compared in (41) the dominance rule is satisfied very rare. In order to overcome this problem, we note the following property of constraint (42)

$$\hat{c}(L) = \overline{c}(L) - \sigma \lfloor m(L) \rfloor = \overline{c}(L) + k\sigma - \sigma \lfloor m(L) - k \rfloor \tag{43}$$

for any integer k. Hence a label $(\hat{c}(L), r(L), m(L))$ is equivalent to a label $(\hat{c}(L) - k\sigma, r(L), m(L) - k)$, meaning that we can make resources comparable in (41) at the cost of modifying $\overline{c}(L)$ in (40) and vice versa. This is the main idea in Proposition 5, 6 and 7 to be presented.

For a label L let

$$\mathcal{T}(L) = \left(\sum_{i \in V(L)} u_i \right) \bmod 1$$

be the amount involved in the cut since the last penalty was paid, i.e., the fractional part of $\sum_{i \in V(L)} u_i$. Recall $\mathcal{E}(L)$ as the set of feasible extensions from the label L to the target node o' and note that when label L_i dominates label L_j, their common extensions are $\mathcal{E}(L_j)$ due to (38). The following cost dominance criteria are obtained for a single CG rank-1 cut:

Proposition 5. *If $\mathcal{T}(L_i) \leq \mathcal{T}(L_j)$, $\overline{v}(L_i) = \overline{v}(L_j)$, $\hat{c}(L_i) \leq \hat{c}(L_j)$, and $r(L_i) \leq r(L_j)$ $\forall r \in R$, then label L_i dominates label L_j.*

Proof. Consider any common extension $\epsilon \in \mathcal{E}(L_j)$. Since $\mathcal{T}(L_i) \leq \mathcal{T}(L_j)$ the relation between the number of future penalties for the two labels when concatenated with ϵ is:

$$\left\lfloor \sum_{i \in \epsilon} u_i + \mathcal{T}(L_i) \right\rfloor \leq \left\lfloor \sum_{i \in \epsilon} u_i + \mathcal{T}(L_j) \right\rfloor$$

This leads to the following relation between the costs:

$$\begin{aligned} & \hat{c}(L_i + \epsilon) = \hat{c}(L_i) + \overline{c}(\epsilon) - \sigma \left\lfloor \sum_{i \in \epsilon} u_i + \mathcal{T}(L_i) \right\rfloor \\ \leq \quad & \hat{c}(L_j) + \overline{c}(\epsilon) - \sigma \left\lfloor \sum_{i \in \epsilon} u_i + \mathcal{T}(L_j) \right\rfloor = \hat{c}(L_j + \epsilon) \end{aligned}$$

Hence, label L_i dominates label L_j. □

Proposition 6. *If $\mathcal{T}(L_i) > \mathcal{T}(L_j)$, $\overline{v}(L_i) = \overline{v}(L_j)$, $\hat{c}(L_i) - \sigma \leq \hat{c}(L_j)$, and $r(L_i) \leq r(L_j)$ $\forall r \in R$, then label L_i dominates label L_j.*

Proof. Consider any common extension $\epsilon \in \mathcal{E}(L_j)$. Since $\mathcal{T}(L_i) > \mathcal{T}(L_j)$ the relation between the number of future penalties for the two labels when concatenated with ϵ is:

$$\left\lfloor \sum_{i \in \epsilon} u_i + \mathcal{T}(L_i) \right\rfloor \geq \left\lfloor \sum_{i \in \epsilon} u_i + \mathcal{T}(L_j) \right\rfloor \tag{44}$$

Since $0 \leq \mathcal{T}(L_j) < \mathcal{T}(L_i) < 1$ it is clear that the left hand side of (44) is at most one unit larger than the right hand side, i.e., label L_i will pay the penalty at most one more time than label L_j. Hence,

$$\left\lfloor \sum_{i \in \epsilon} u_i + \mathcal{T}(L_i) \right\rfloor - 1 \leq \left\lfloor \sum_{i \in \epsilon} u_i + \mathcal{T}(L_j) \right\rfloor$$

That is, the additional cost of extending L_i with ϵ is at most $-\sigma$ more than extending L_j with ϵ. This leads to the following relation between the costs:

$$\begin{aligned}
\hat{c}(L_i + \epsilon) &= \hat{c}(L_i) + \overline{c}(\epsilon) - \sigma \left\lfloor \sum_{i \in \epsilon} u_i + \mathcal{T}(L_i) \right\rfloor \\
&= \hat{c}(L_i) - \sigma + \overline{c}(\epsilon) - \sigma \left(\left\lfloor \sum_{i \in \epsilon} u_i + \mathcal{T}(L_i) \right\rfloor - 1 \right) \\
&\leq \hat{c}(L_j) + \overline{c}(\epsilon) - \sigma \left\lfloor \sum_{i \in \epsilon} u_i + \mathcal{T}(L_j) \right\rfloor \\
&= \hat{c}(L_j + \epsilon)
\end{aligned}$$

Hence label L_i dominates label L_j. □

Observe that if $\mathcal{T}(L_i) + \sum_{i \in \epsilon} u_i < 1$ for all $\epsilon \in \mathcal{E}(L_j)$, it is not possible to trigger a penalty, i.e., the temporary penalty to the cost of L_i can be disregarded.

In case of several CG rank-1 cuts, the new dominance criteria are as follows:

Proposition 7. *Let $Q = \{q : \sigma_q < 0 \wedge \mathcal{T}_q(L_i) > \mathcal{T}_q(L_j)\}$. Then label L_i dominates label L_j if:*

$$\overline{v}(L_i) = \overline{v}(L_j) \tag{45}$$

$$\hat{c}(L_i) - \sum_{q \in Q} \sigma_q \leq \hat{c}(L_j) \tag{46}$$

$$r(L_i) \leq r(L_j) \qquad \forall r \in R \tag{47}$$

Proof. The validity of (46) follows directly from Propositions 5 and 6. The validity of (45) and (47) follows from Proposition 4. □

5 Experimental Results

The experimental study is intended to show how much it is possible to strengthen the lower bound by adding CG rank-1 cuts, while still being able to solve the corresponding pricing problem in reasonable time. The SR inequalities have already proved their worth, see Jepsen et al. [23], Desaulniers et al. [9], but in both cases only sets of rows with size 3 were included, i.e., CG rank-1 cuts with precisely 3 non-zero entries in the CG multiplier vector. Hence, it is expected that the introduction of a separation routine for denser CG multiplier vectors could improve the lower bounds further. Using the exact separation routine for the CG rank-1 cuts is expected to be time consuming, but for test purposes it is interesting to see how well the column generation reacts to these cuts and also how much the lower bounds are improved.

5.1 Settings

The test instances are the well known benchmarks introduced by Solomon [33]. The benchmarks are divided into two series, both of which are again divided into a C (customers are grouped in larger clusters), an R (customers are distributed randomly), and an RC (a mix of the two previous) series. Of the 56 instances with 100 customers five instances are unsolved at the time of writing. Furthermore, 16 of the solved instances have not yet been solved in the root node of the branch-and-bound tree. We will only consider the R and RC instances, since all C instances can be solved in the root node without cutting planes, see Jepsen et al. [23], Desaulniers et al. [9].

The experiments were performed on a Pentium 4 3.0 GHz with 1 GB RAM. The basic BCP algorithm was developed with the framework COIN, see Lougee-Heimer [27]. The exact MIP-based CG rank-1 separation procedure is a slight modified version of a procedure provided by Hunsaker [19]. The MIPs were solved using CPLEX 9.1 with a maximal running time of 3600 seconds.

An exact separation procedure for a limited set of the SR inequalities have been developed exploiting the SSE2 vector-processing instructions intended for multimedia floating-point purposes which are present in all x86 processors since the introduction of Pentium 4 in 2001. The separation routine is an exact enumeration of SR inequalities with multipliers $u_i \in \{0, \frac{1}{k}, \ldots, \frac{k-1}{k}\}$ for $i \in C$ where $\sum_{i \in C} u_i = \frac{n}{k}$, and $0 < k < n \leq |C|$ and k and n are integer, i.e., mod-k cuts with restriction on the sum of the multipliers.

Our implementation of the brute-force evaluation of all sub-multisets of rows of size n, can evaluate the SR inequalities (33) in constant time for each sub-multiset using the vector-processing capabilities. This makes it possible to separate all violated cuts in time $|S|^n/n!$ when $|P| \leq 16$, where S is the set of rows and P is the set of basis columns. Still, the complexity is so high that we cannot expect to separate inequalities with more than seven non-zero coefficients in reasonably time.

Note that our implementation of the BCP algorithm is not competitive with the recent implementation by Desaulniers et al. [9]. Also it is slower than the one used in Jepsen et al. [23] due to the implementation of the more general dominance criteria in the label algorithm. However, the point of our experiments is to study the quality of the lower bounds, i.e., the number of branch nodes, compared to the increase in computational time of the pricing problem by adding various cuts. These conclusions hold for all implementations based on the same decomposition.

5.2 Lower Bounds

Table 1 and 2 show the lower bounds obtained in the root node when different cutting policies are applied.

The cutting policies are:

"no"	No cutting planes
"$n = 3$"	SR cuts with $n = 3$ and $k = 2$
"$n \leq 5$"	Like option $n = 3$ and with $n = 5$ and $k = 2, 3$
"$n \leq 7$"	Like option $n \leq 5$ and with $n = 7$ and $k = 2, 3, 4$
"CG1"	General CG rank-1 cuts

A maximum of 50 cuts violating more than 0.0001 are added in each iteration. No time limit was imposed, but the space limit of 1 GB RAM prevented some instances to run to

Table 1: Lower bound comparison for the 1-series.

Instance	no	$n = 3$	$n \leq 5$	$n \leq 7$	CG1	UB
R101	1631.2	1634.0	1636.3	1636.3	**1637.7**	**1637.7**
R102	**1466.6**	**1466.6**	**1466.6**	**1466.6**	**1466.6**	**1466.6**
R103	1206.8	**1208.7**	**1208.7**	**1208.7**	**1208.7**	**1208.7**
R104	956.9	971.3	**971.5**	**971.5**	**971.5**	**971.5**
R105	1346.2	1355.2	**1355.3**	**1355.3**	**1355.3**	**1355.3**
R106	1227.0	**1234.6**	**1234.6**	**1234.6**	**1234.6**	**1234.6**
R107	1053.3	1064.3	**1064.6**	**1064.6**	**1064.6**	**1064.6**
R108	913.6	**932.1**	**932.1**	**932.1**	**932.1**	**932.1**
R109	1134.3	1144.1	1146.7	**1146.9**	**1146.9**	**1146.9**
R110	1055.6	**1068.0**	**1068.0**	**1068.0**	**1068.0**	**1068.0**
R111	1034.8	1045.9	1047.3	-	-	**1048.7**
R112	926.8	943.5	-	-	-	**948.6**
RC101	1584.1	**1619.8**	**1619.8**	**1619.8**	**1619.8**	**1619.8**
RC102	1406.3	**1457.4**	**1457.4**	**1457.4**	**1457.4**	**1457.4**
RC103	1225.6	1257.7	**1258.0**	**1258.0**	**1258.0**	**1258.0**
RC104	1101.9	1129.9	-	-	-	**1132.3**
RC105	1472.0	**1513.7**	**1513.7**	**1513.7**	**1513.7**	**1513.7**
RC106	1318.8	1367.3	1371.9	**1372.7**	**1372.7**	**1372.7**
RC107	1183.4	**1207.8**	**1207.8**	**1207.8**	**1207.8**	**1207.8**
RC108	1073.5	**1114.2**	**1114.2**	**1114.2**	**1114.2**	**1114.2**

completion.

Upper bounds in the "UB" column are optimal values or best known upper bounds. Entries in tables marked with an asterisk * are from Danna and Le Pape [7], entries marked with a double-asterisk ** are from Desaulniers et al. [9], and entries marked with a triple-asterisk *** are from Jepsen et al. [23]. A dash indicates that our implementation failed due to memory limitation. Entries in bold face indicate optimal integer solution.

Of the 28 solved instances one instance (R102) was solved without adding any cuts. The lower bounds for all remaining instances were considerably improved by adding "$n = 3$" cuts resulting in integer solutions for 15 of the 27 remaining (17 out of 33 when considering the results of Desaulniers et al. [9]). When adding "$n \leq 5$" cuts improvements were present in all but one instance (RC201) resulting in further five integer solutions of the 10 remaining instances that could be solved with this approach. Of the remaining four instances solved with "$n \leq 7$" cuts, two showed no improvement and two resulted in integer solutions. The last two instances were solved to integrality when applying CG rank-1 cuts. Hence, we succeeded in closing the gap between the upper and lower bound for all the instances that we were able to solve within the memory limit.

Tables 1 and 2 also show that the SR inequalities provide almost as good lower bounds as general CG rank-1 cuts. For "$n = 7$" the SR inequalities become time consuming to separate, and hence in practical applications one should confine to "$n = 3$" or "$n \leq 5$".

Table 3 presents an overview of problems solved in the root node as reported in this chapter or by Jepsen et al. [23] or Desaulniers et al. [9]. Column "solved" refers to the number of instances solved to optimality at the time of writing and "total" refers to the total number of instances. Results from the C-series are included for completeness.

As already noted, adding SR inequalities and CG rank-1 cuts greatly strengthens the

Table 2: Lower bound comparison for the 2-series.

Instance	no	$n = 3$	$n \leq 5$	$n \leq 7$	CG1	UB
R201	1140.3	**1143.2**	**1143.2**	**1143.2**	**1143.2**	**1143.2**
R202	1022.3	1027.3	**1029.6**	**1029.6**	**1029.6**	**1029.6**
R203	867.0	**870.8**	**870.8**	**870.8**	**870.8**	**870.8**
R204	-	-	-	-	-	**731.3
R205	939.0	-	-	-	-	**949.8**
R206	866.9	****875.9**	-	-	-	**875.9**
R207	**790.7	****794.0**	-	-	-	**794.0**
R208	-	-	-	-	-	*701.2
R209	841.5	***854.8	-	-	-	**854.8**
R210	889.4	-	-	-	-	**900.5**
R211	-	-	-	-	-	**746.7
RC201	1256.0	1261.7	1261.7	1261.7	**1261.8**	**1261.8**
RC202	1088.1	**1092.3**	**1092.3**	**1092.3**	**1092.3**	**1092.3**
RC203	922.6	**923.7**	**923.7**	**923.7**	**923.7**	**923.7**
RC204	-	-	-	-	-	*783.5
RC205	1147.7	**1154.0**	**1154.0**	**1154.0**	**1154.0**	**1154.0**
RC206	1038.6	**1051.1**	**1051.1**	**1051.1**	**1051.1**	**1051.1**
RC207	947.4	-	-	-	-	**962.9**
RC208	-	-	-	-	-	**776.5

Table 3: Summary of instances solved in the root node.

Instance	no	$n = 3$	$n \leq 5$	$n \leq 7$	CG1	solved	total
C1	9	9	9	9	9	9	9
C2	8	8	8	8	8	8	8
R1	1	5	8	9	10	12	12
R2	0	4	5	5	5	8	11
RC1	0	5	6	7	7	8	8
RC2	0	4	4	4	5	6	8
All	18	35	40	42	44	51	56

lower bounds. Of the 56 instances 35 were previously reported solved in the root node by Jepsen et al. [23], Desaulniers et al. [9]. With our additional cutting planes we were able to solve an additional nine instances in the root node of the remaining 16 previously solved instances. Note that all the instances we were able to solve were solved in the root node. The remaining seven instances, which have previously been solved with "$n = 3$", could not be solved with the current implementation due to hardware limitations. Hence, there exists 12 Solomon instances (seven solved with branching and five unsolved) where CG rank-1 cuts could potentially improve the lower bound in the root node.

5.3 Running Times of the Pricing Problem

Table 4 and 5 contain the results obtained when solving the instances to optimality using different cutting planes. In column "CPU" we report the CPU-time in seconds for solving the last pricing problem, while column "cuts" gives the number of cuts applied. Column "BB" indicates the number of branch-and-bound nodes considered. As before, a dash in the tables

indicates that a memory insufficiency had occurred. Entries marked with a double-asterisk ** are from Desaulniers et al. [9].

Table 4: Running time for pricing problem and number of branch-and-bound nodes for the 1-series. [1)] Data log-files were lost during machine upgrade.

	no		$n = 3$			$n \leq 5$			CG1		
Instance	CPU	BB	CPU	cuts	BB	CPU	cuts	BB	CPU	cuts	BB
R101	0.1	11	0.1	2	3	0.1	4	3	0.1	15	1
R102	0.2	1	0.2	0	1	0.2	0	1	0.2	0	1
R103	0.4	15	1.3	33	1	1.3	33	1	1.3	33	1
R104	5.8	-	910.5	328	3	-	-	1^1	-	-	1
R105	0.1	55	0.2	52	3	0.2	56	1	0.2	56	1
R106	0.5	147	4.8	114	1	4.8	114	1	4.8	114	1
R107	2.2	-	46.1	224	3	78.4	242	1	78.4	242	1
R108	13.0	-	244.8	192	1	244.8	192	1	244.8	192	1
R109	0.3	-	1.6	127	17	8.7	374	3	10.0	367	1
R110	1.1	-	26.0	269	1	26.0	269	1	26.0	269	1
R111	1.5	-	36.6	175	39	293.7	379	-	-	-	-
R112	35.9	-	-	-	9^1	-	-	-	-	-	-
RC101	0.1	59	0.2	69	1	0.2	69	1	0.2	69	1
RC102	0.3	-	1.4	140	1	1.4	140	1	1.4	140	1
RC103	1.2	-	42.8	276	3	49.1	290	1	49.1	290	1
RC104	15.6	-	569.2	237	7	-	-	-	-	-	-
RC105	0.2	191	0.5	73	1	0.5	73	1	0.5	73	1
RC106	0.3	-	3.5	250	37	16.5	543	5	21.6	572	1
RC107	1.4	-	4.3	85	1	4.3	85	1	4.3	85	1
RC108	9.7	-	86.7	175	1	86.7	175	1	86.7	175	1

The tables show that adding "$n \leq 5$" cuts and "CG1" cuts is relatively cheap with respect to the running time of the pricing problem, while decreasing the number of branch-and-bound nodes significantly e.g., in instances R109, RC106, and R202.

If we had access to "ideal" heuristics for the pricing problem (with low running time and high solution quality) we would only need to solve one pricing problem to optimality in each branch-and-bound node. The running time of the overall algorithm would then be dictated by the running time for optimally solving the pricing (CPU) and the number of branch-and-bound nodes (BB). With the exception of R202 (where massive paging occurred due to lack of memory) the lower bound on the running time "BB $\times$ CPU" is not increasing when n grows and "CG1" cuts are applied. This shows, that good heuristics for the pricing problem can make the addition of SR and CG-1 cuts attractive for the overall running time.

6 Concluding Remarks

We have demonstrated that it is possible to apply general CG rank-1 cuts derived from the master problem formulation in a BCP algorithm for VRPTW. As each cut results in the introduction of a new resource in the pricing problem it was necessary to develop new, tighter dominance rules for use in the pricing algorithm.

Our computational experiments indicate that the addition of CG rank-1 cuts leads to significantly improved lower bounds. In our tests the cuts made it possible to close the gap

Table 5: Running time for pricing problem and number of branch-and-bound nodes for the 2-series.

	no		$n = 3$			$n \leq 5$			CG1		
Instance	CPU	BB	CPU	cuts	BB	CPU	cuts	BB	CPU	cuts	BB
R201	0.2	-	0.4	15	1	0.4	15	1	0.4	15	1
R202	2.9	-	3.0	24	13	419.6	132	1	419.6	132	1
R203	83.2	-	505.6	35	1	505.6	35	1	505.6	35	1
R204	-	-	-	-	-	-	-	-	-	-	-
R205	1.5	-	-	**345	**9	-	-	-	-	-	-
R206	131.7	-	-	**171	**1	-	-	-	-	-	-
R207	-	-	-	**24	**1	-	-	-	-	-	-
R208	-	-	-	-	-	-	-	-	-	-	-
R209	6.5	-	-	**248	**3	-	-	-	-	-	-
R210	-	-	-	**266	**5	-	-	-	-	-	-
R211	-	-	-	-	-	-	-	-	-	-	-
RC201	0.2	-	0.3	25	3	0.3	25	3	0.3	29	1
RC202	0.6	-	1.7	26	1	1.7	26	1	1.7	26	1
RC203	58.8	11	185.2	15	1	185.2	15	1	185.2	15	1
RC204	-	-	-	-	-	-	-	-	-	-	-
RC205	1.0	-	1.8	21	1	1.8	21	1	1.8	21	1
RC206	1.7	-	4.6	23	1	4.6	23	1	4.6	23	1
RC207	-	-	-	**210	**5	-	-	-	-	-	-
RC208	-	-	-	-	-	-	-	-	-	-	-

between the upper and lower bounds in the root node of the branch-and-bound tree for 44 of the 51 currently solvable instances from Solomon's test library. This is an additional 9 instances compared to previous results. The increased complexity of the pricing problem caused by CG rank-1 cuts do affect the running time of the pricing problems but not significantly.

This indicates that CG rank-1 inequalities may be essential when solving larger instances to optimality, as one cannot expect that the branching process will close the gap between the upper and lower bound in reasonable time. Note that one should also take into account the additional time spent in each branch node since the number of LP iterations increases when valid inequalities are added. As for classical branch-and-cut algorithms it will always be a question when to add cuts and when to start branching.

Another important note is the separation time of the CG rank-1 cuts which can indeed be very time consuming. Also the current MIP-based heuristics only finds a limited number of violated cuts as the main effort is put in cut violation quality not violated cut quantity. We suggest that MIP-based heuristics which focus on finding numerous violated CG rank-1 cuts could improve the performance of the BCP algorithm. Fortunately the SR inequalities generally give rise to almost as tight lower bounds as general CG rank-1 cuts, while being easier to handle in the pricing problem (due to integer modulo operations, see Jepsen et al. [23]). For $n = 7$ the separation of SR inequalities takes almost one hour, making them too expensive to separate. For $n \leq 5$ the inequalities can be separated in a couple of minutes. So until more efficient separation methods have been developed, one should only apply SR inequalities for $n \leq 5$.

During our experiments we noticed that specific values of the CG multipliers u occurred more frequently than others. For instance, multiplier vectors $\mathbf{u} \in \{0, \frac{1}{2}\}^{|C|}$ occurred very

frequently, showing that it is promising to investigate these inequalities further (note that the SR inequalities for a given n with $k = 2$ are a subset of these inequalities). Knowing the structure of promising CG rank-1 inequalities will make it possible to develop fast, specialized separation heuristics and better handling of these specific inequalities in the pricing problem. Adapting the separation algorithm by Caprara et al. [3] for maximally violated mod-k cuts in the master problem could be an interesting direction of research.

References

[1] R. E. Bixby, S. Ceria, C. M. McZeal, and M. W. P Savelsbergh. An updated mixed integer programming library: MIPLIB 3.0. *Optima*, 58:12–15, 1998.

[2] N. Boland, J. Dethridge, and I. Dumitrescu. Accelerated label setting algorithms for the elementary resource constrained shortest path problem. *Operation Research Letters*, 34 (1):58–68, 2006. doi: 10.1016/j.orl.2004.11.011.

[3] A. Caprara, M. Fischetti, and A. N. Letchford. On the separation of maximally violated mod-k cuts. *Mathematical Programming*, 87(A):37–56, 1999. doi: 10.1007/s101079900107.

[4] A. Chabrier. Vehicle routing problem with elementary shortest path based column generation. *Computers & Operations Research*, 33(10):2972–2990, 2006. doi: 10.1016/j.cor.2005.02.029.

[5] V. Chvatal. Edmonds polytopes and hierarchy of combinatorial problems. *Discrete Mathematics*, 4(4):305–337, 1973. doi: 10.1016/0012-365X(73)90167-2.

[6] W. Cook and J. L. Rich. A parallel cutting plane algorithm for the vehicle routing problem with time windows. Technical Report TR99-04, Computational and Applied Mathematics, Rice University, Houston, Texas, USA, 1999.

[7] E. Danna and C. Le Pape. Branch-and-price heuristics: A case study on the vehicle routing problem with time windows. In G. Desaulniers, J. Desrosiers, and M. M. Solomon, editors, *Column Generation*, chapter 4, pages 99–129. Springer, 2005. doi: 10.1007/0-387-25486-2_4.

[8] G. Desaulniers, J. Desrosiers, J. Ioachim, I. M. Solomon, F. Soumis, and D. Villeneuve. A unified framework for deterministic time constrained vehicle routing and crew scheduling problems. In T. G. Crainic and G. Laporte, editors, *Fleet Management and Logistics*, pages 57–93. Kluwer, 1998.

[9] G. Desaulniers, F. Lessard, and A. Hadjar. Tabu search, partial elementarity, and generalized k-path inequalities for the vehicle routing problem with time windows. *Transportation Science*, 42(3):387–404, 2008. doi: 10.1287/trsc.1070.0223.

[10] M. Desrochers, J. Desrosiers, and M. Solomon. A new optimization algorithm for the vehicle routing problem with time windows. *Operations Research*, 40(2):342–354, 1992. doi: 10.1287/opre.40.2.342.

[11] M. Dror. Note on the complexity of the shortest path models for column generation in VRPTW. *Operations Research*, 42(5):977–979, 1994. doi: 10.1287/opre.42.5.977.

[12] I. Dumitrescu. *Constrained Path and Cycle Problems*. PhD thesis, Department of Mathematics and Statistics, University of Melbourne, Australia, 2002.

[13] F. Eisenbrand. Note - on the membership problem for the elementary closure of a polyhedron. *Combinatorica*, 19(2):297–300, 1999. doi: 10.1007/s004930050057.

[14] D. Feillet, P. Dejax, M. Gendreau, and C. Gueguen. An exact algorithm for the elementary shortest path problem with resource constraints: Application to some vehicle routing problems. *Networks*, 44(3):216–229, 2004. doi: 10.1002/net.v44:3.

[15] M. Fischetti and A. Lodi. Optimizing over the first Chvátal closure. *Mathematical Programming*, 110(1):3–20, 2006. doi: 10.1007/s10107-006-0054-8.

[16] R. Fukasawa, H. Longo, J. Lysgaard, M. Poggi de AragÃ£o, M. Reis, E. Uchoa, and R.F. Werneck. Robust branch-and-cut-and-price for the capacitated vehicle routing problem. *Mathematical Programming*, 106(3):491–511, 2006. doi: 10.1007/s10107-005-0644-x.

[17] R.E. Gomory. Outline of an algorithm for integer solutions to linear programs. *Bulletin of the AMS*, 64:275–278, 1958. doi: 10.1090/S0002-9904-1958-10224-4.

[18] R.E. Gomory. An algorithm for integer solutions to linear programs. In R.L. Graves and P. Wolfe, editors, *Recent Advances in Mathematical Programming*, pages 269–302. McGraw-Hill, New York, 1963.

[19] B. Hunsaker. Cg-rank. http://www.rosemaryroad.org/brady/cg-rank/index.html, 2005.

[20] S. Irnich. Resource extension functions: Properties, inversion, and generalization to segments. *OR Spectrum*, 30(1):113–148, 2008. doi: 10.1007/s00291-007-0083-6.

[21] S. Irnich and G. Desaulniers. Shortest path problems with resource constraints. In G. Desaulniers, Jacques Desrosiers, and M.M. Solomon, editors, *Column Generation*, chapter 2, pages 33–65. Springer, 2005. doi: 10.1007/0-387-25486-2_2.

[22] S. Irnich and D. Villeneuve. The shortest path problem with resource constraints and k-cycle elimination for $k \geq 3$. *INFORMS Journal on Computing*, 18:391–406, 2006. doi: 10.1287/ijoc.1040.0117.

[23] M. Jepsen, B. Petersen, S. Spoorendonk, and D. Pisinger. Subset-row inequalities applied to the vehicle-routing problem with time windows. *Operations Research*, 56(2):497–511, 2008. doi: 10.1287/opre.1070.0449.

[24] B. Kallehauge, J. Larsen, and O.B.G. Madsen. Lagrangean duality and non-differentiable optimization applied on routing with time windows - experimental results. Technical Report Internal report IMM-REP-2000-8, Department of Mathematical Modelling, Technical University of Denmark, Lyngby, Denmark, 2000.

[25] N. Kohl, J. Desrosiers, O. B. G. Madsen, M. M. Solomon, and F. Soumis. 2-path cuts for the vehicle routing problem with time windows. *Transportation Science*, 33(1):101–116, 1999. doi: 10.1287/trsc.33.1.101.

[26] J. Larsen. *Parallelization of the vehicle routing problem with time windows.* PhD thesis, Department of Mathematical Modelling, Technical University of Denmark, Lyngby, Denmark, 1999.

[27] Robin Lougee-Heimer. The Common Optimization INterface for Operations Research: Promoting open-source software in the operations research community. *IBM Journal of Research and Development*, 47(1):57–66, 2003. doi: 10.1147/rd.471.0057.

[28] G. Nemhauser and S. Park. A polyhedral approach to edge coloring. *Operations Research Letters*, 10(6):315–322, 1991. doi: 10.1016/0167-6377(91)90003-8.

[29] G. Righini and M. Salani. Symmetry helps: bounded bi-directional dynamic programming for the elementary shortest path problem with resource constraints. *Discrete Optimization*, 3(3):255–273, 2006. doi: 10.1016/j.disopt.2006.05.007.

[30] G. Righini and M. Salani. New dynamic programming algorithms for the resource constrained shortest path problem. *Networks*, 51(3):155–170., 2008. doi: 10.1002/net.20212.

[31] M. Salani. *Branch-and-Price Algorithms for Vehicle Routing Problems.* PhD thesis, Universitá Degli Studi Di Milano, Facoltá di Scienza Matematiche, Fisuche e Naturali Dipartimento di Technologie dell'Informazione, Milano, Italy, 2005.

[32] A. Schrijver. On cutting planes. *Annals of Discrete Mathematics*, 9:291–296, 1980. doi: 10.1016/S0167-5060(08)70085-2.

[33] M. M. Solomon. Algorithms for the vehicle routing and scheduling problem with time window constraints. *Operations Research*, 35(2):234–265, 1987. doi: 10.1287/opre.35.2.254.

[34] P. Toth and D. Vigo. An overview of vehicle routing problems. In P. Toth and D. Vigo, editors, *The Vehicle Routing Problem*, chapter 1, pages 1–26. SIAM, 2002.

Chapter 5

A Branch-and-Cut Algorithm for the Elementary Shortest Path Problem with Resource Constraints

Mads Jepsen
DIKU Department of Computer Science, University of Copenhagen

Bjørn Petersen
DIKU Department of Computer Science, University of Copenhagen

Simon Spoorendonk
DIKU Department of Computer Science, University of Copenhagen

Abstract

This paper introduces a branch-and-cut (BAC) algorithm for the elementary shortest path problem with resource constraints (ESPPRC), which commonly appears as a subproblem in column generation based algorithms, e.g., in the classical Dantzig-Wolfe decomposition of the capacitated vehicle routing problem. Specifically, we consider an undirected graph with arbitrary edge costs (i.e., negative cost cycles may appear) and with resources that are equally constrained at all nodes and arcs. A mathematical model and valid inequalities are presented, including a new family of valid inequalities denoted the generalized capacity inequalities. Experimental tests are performed on a set of generated instances with graphs of high edge density and a set of instances from the literature. Traditionally, labeling algorithms have been the dominant solution method for the ESPPRC, but experimental results show that the BAC algorithm is superior on all the tested instances.

Keywords: Branch-and-cut algorithm, elementary shortest path problem with resource constraints

In revision.

1 Introduction

The elementary shortest path problem with resource constraints (ESPPRC) can informally be stated as the problem of finding a shortest path between two nodes in a graph where resources are accumulated along the path, and where the amount of resources are constrained.

In this paper, we consider the case where the graph is undirected and edge costs are allowed to take on any value. Furthermore, we demand that the path is simple such that no nodes are visited more than once. The resources considered in this paper are all bounded such that the lower and upper bound of the amount of a resource that are accumulated along the path is equal for all nodes and edges. We assume, that the resource lower bounds are zero and that the accumulations are monotone increasing and only performed at the nodes. This type of *globally constrained* resource compares to the vehicle capacity known from the capacitated vehicle routing problem, where the resource accumulates a positive value (demand) at each node and the upper bound (capacity of the vehicle) may not be exceeded.

It is now possible to give a more formal statement of the ESPPRC. Let $G = (V, E)$ be an undirected graph with nodes V and edges E. Let a cost c_e be associated with each edge $e \in E$, let d_i^r be a positive resource accumulation associated to each node $i \in V$ for each resource $r \in R$, and let Q^r be the upper bound on the resource r. Then given a source node $s \in V$ and a target node $t \in V$; find a path between s and t with minimum cost satisfying that the sum of the resource r from at each of the visited nodes is not more than Q^r for all $r \in R$.

The ESPPRC defined as above is $\mathcal{NP}$-hard in the strong sense. This is easily verified by reduction from the longest path problem. The definition of the ESPPRC varies in the literature, especially with regard to edge costs, resource bounds, and resource accumulations.

Beasley and Christofides [7] presented a mathematical model (very similar to the one used in this paper) and performed experimental tests using a branch-and-bound algorithm based on Lagrangian dual bounds. Dumitrescu and Boland [15] presented a labeling algorithm that was improved by preprocessing based on resource availability. Carlyle et al. [10] proposed a Lagrangian relaxation algorithm where paths with cost between the Lagrangian bound and the current upper bound are found using the k-shortest path algorithm by Carlyle and Wood [9]. Common for these approaches are that they all assume that the graph have no negative cost cycles. This makes it easier to ensure simplicity of the path, since it cannot pay off to visit a node more than once. The ESPPRC in this form is weakly $\mathcal{NP}$-hard, and results of the algorithms presented above are therefore not directly comparable to the results in this paper.

Another common definition is to consider resource bounds individually for each node (or edge). In this case, it is often necessary to consider an undirected graph, because the direction of the path determines the correct resource accumulation at a given node. Such resources compare to the time in the vehicle routing problem with time windows, where the resource (time) accumulates for each edge and the nodes must be visited within a resource window (a time window defined by a minimum and a maximum arrival time for a node). Such resources are said to be *locally constrained*. Dror [14] proved that the ESPPRC with a single globally constrained resource and a single locally constrained resource is $\mathcal{NP}$-hard in the strong sense. Feillet et al. [16] presented a labeling algorithm where the simplicity of the path is ensured with the use of an additional globally constrained resource per node. Chabrier [11] improved on the labeling algorithm by applying various bounding procedures to avoid extending unwanted paths. Righini and Salani [24] proposed a bi-directional labeling algorithm where paths are

extended from both the source node and the target node until a given *middle* of a monotone increasing resource is reached, e.g., when half the time was consumed on the path. The partial paths are then combined to construct a full path. Independently, Boland et al. [8] and Righini and Salani [25] proposed to extend the labeling algorithm by relaxing the node resources and adding them incrementally until the path is simple. In the former paper, this is referred to as a *state space augmentation* algorithm, and in the latter, it is denoted a *decremental state space relaxation* algorithm. Furthermore, Righini and Salani [25] propose to use the result of the relaxed problem in a branch-and-bound algorithm.

The algorithms presented above are mainly labeling algorithms. As mentioned in Beasley and Christofides [7], even the algorithms based on Lagrangian relaxation make use of a dynamic programming algorithm if negative costs cycles are allowed. The strength of the labeling algorithms is, that the locally constrained resources are easily implemented, since the paths are build piece by piece such that resource limits can be checked at every step. In fact, non-linear functions for accumulation of resources can be handled easily, see e.g., Desaulniers et al. [12]. Generally, labeling algorithms are assumed to perform well on a sparse graphs with tightly constrained resources, since this yields a very reduced solution space to search, i.e., few states in the dynamic programming table needs to be searched. However, when the graph is dense and the resources are loosely constrained, the labeling algorithms get closer to a full enumeration of all paths.

Modeling of resources (accumulation and bounds) is limited in branch-and-cut (BAC) algorithms that are based on linear programming (which is the case in this paper). Globally constrained resources with positive accumulation can be modeled as single knapsack constraints (and remain simple to model with negative accumulation). Locally constrained resources with positive accumulation can be modeled for a directed graph with the use of the Miller-Tucker-Zemlin (MTZ) constraints, see Miller et al. [22]. This gives rise to $|E|$ additional constraints and $|V|$ variables per resource. Another modeling approach gives rise to $|V|$ constraints and $|E|$ variables per resource, see e.g., Ascheuer et al. [1, 2]. A different approach is to relax the resource constraints and, in a cutting plane fashion, make use of the infeasible path inequalities which cuts of any path (or partial path) that violates a resource bound. In Ascheuer et al. [2] a BAC algorithm for the traveling salesman problem with time windows makes use of the three modeling approaches described above. Results indicate that the infeasible path inequalities are to be preferred.

When considering the ESPPRC as a subproblem in a column generation context, another issue comes up. Recent branch-and-cut-and-price algorithms, see e.g., Jepsen et al. [20], Petersen et al. [23], Desaulniers et al. [13], Spoorendonk and Desaulniers [27], Baldacci et al. [5], make use of cutting planes where the dual values are not directly subtractable from the edge costs, which has previously been the preferred approach, see e.g., Fukasawa et al. [18]. The subtraction of such dual values depend on the complete path and can be very cumbersome to overcome in labeling algorithms. However, when following the ideas in Spoorendonk et al. [28] it is clarified how to model the additional costs in the subproblem, whereupon the BAC algorithm can be applied.

Results by Ascheuer et al. [2] for the traveling salesman problem with time windows indicate, that it is expensive (in running time) in a BAC algorithm, to use either of the modeling approaches for locally constrained resources, i.e., the time windows. However, when only globally constrained resources are considered, it seem likely that a BAC algorithm can be competitive with labeling algorithms. So, although locally constrained resources can be modeled in a BAC algorithm, it is not within the scope of this paper to investigate

that approach. The reason for considering an undirected graph in this paper is mainly for simplicity. The BAC algorithm can easily be extended to the directed case by doubling the number of variables in the mathematical formulation. Neither of the separation routines are affected by this (except for the doubling of variables). the undirected graphs favors the

The main contribution of this paper is the introduction of a BAC algorithm for solving the ESPPRC. This includes a 2-index mathematical model and a presentation of valid inequalities with emphasis on the introduction of the generalized capacity inequalities. The computational results indicate that the BAC algorithm is competitive with labeling algorithms when considering dense graphs, and even more so when the resources are loosely constrained.

The paper is outlined as follows: Section 2 presents work on BAC algorithms for problems that are related to the ESPPRC and Section 3 contains a formal integer programming model of the ESPPRC. Section 4 describes the cutting planes used in the BAC algorithm and the computational results are found in Section 5. Section 6 holds concluding remarks and suggestions for further research.

2 Related Work

Bauer et al. [6] suggested to solve the ESPPRC by a BAC algorithm, but to our knowledge nothing further has been published in the literature, although several BAC algorithms exist for problems related to the ESPPRC. Bauer et al. [6] consider the knapsack constrained circuit problem (KCCP) where a minimal capacitated cycle in a graph is sought. This is equivalent to the ESPPRC if one node is fixed in the KCCP, since this node can be spilt into a source and a target node in the ESPPRC. A BAC algorithm was implemented to solve the KCCP where the demand of the nodes was given with unit weights. This variant is denoted the cardinality constraint circuit problem. The instances considered by Bauer et al. [6] have positive edge costs, but negative cost cycles would not affect the algorithm.

In the prize collecting traveling salesman problem (PCTSP), see e.g., Balas [3, 4], a prize is collected at each visited node and a minimum amount of accumulated prizes must be collected on the tour. That is, the edge costs are positive but the prizes may yield an overall negative solution value. The difference with this variant of the TSP and the ESPPRC is, that in the PCTSP a minimum amount of prizes need to be collected, which forces some of the intermediate nodes to be visited. This is not the case for the ESPPRC as defined in this paper.

In the orienteering problem, see e.g., Fischetti et al. [17], the profit of visiting the nodes is maximized and the length of the tour is bounded by a maximum length. The only difference compared to the definition of the ESPPRC of this paper is, that the resource accumulation is on the edges instead of in the nodes. The instances considered by Fischetti et al. [17] have positive edge costs, but again negative cost cycles would not affect the algorithm.

3 Mathematical Models

This section presents a flow model for the ESPPRC in the undirected graph G. Recall the resource demand d_i^r for nodes $i \in V$, and the resource upper bound Q^r for resource $r \in R$. Let the binary variable x_e indicate the flow on edge $e \in E$. When describing the model some shorthand notation will be used. For a set of nodes $S \subseteq V$ let the set

of edges $\delta(S) = \{(i,j) : i \in S \wedge j \in V \setminus S\}$ denote the edges between S and $V \setminus S$ where $\delta(i)$ is shorthand for $\delta(\{i\})$ when the node set S consists of a single node $i \in V$. Let $E(S) = \{(i,j) : i \in S \wedge j \in S\}$ be the set of edges between the nodes in S. Let the short-hand notation

$$x(T) = \sum_{e \in T} x_e$$

indicate the flow in the edge set T. Let the shorthand notation $y_i = \sum_{e \in \delta(i)} x_e/2$ indicate the flow in node $i \in V \setminus \{s,t\}$, and for a set of nodes $S \subseteq V$ let

$$y(S) = \sum_{i \in S} y_i$$

be the flow in that node set. The mathematical model of the ESPPRC is then:

$$\min \sum_{e \in E} c_e x_e \tag{1}$$

$$\text{s.t. } x(\delta(s)) = 1 \tag{2}$$

$$x(\delta(t)) = 1 \tag{3}$$

$$x(\delta(i)) = 2y_i \qquad i \in V \setminus \{s,t\} \tag{4}$$

$$\sum_{i \in V} d_i^r y_i \leq Q^r \qquad r \in R \tag{5}$$

$$x(E(S)) \leq y(S) - y_i \qquad i \in S, S \subset V, |S| \geq 2 \tag{6}$$

$$x_e \in \{0,1\} \qquad e \in E \tag{7}$$

The objective function (1) minimizes the overall edge cost. Constraints (2) and (3) ensure that the source node and the target node are end points of the path. Constraints (4) are the flow conservation constraints. Constraints (5) impose the resource constraints. Constraints (6) impose connectivity and subtour elimination. Finally, constraints (7) define the domain of the variables. Note, that $y_i \in \{0,1\}$ due to (2), (3), (6), and (7).

This model has $|E|+|V-2|$ variables and an exponential number of constraints due to (6). In a BAC algorithm, these constraints will be relaxed and separated when violated to ensure feasibility. That is, when disregarding constraints (6) the model have $|V| + |R|$ constraints.

4 Cutting Planes

This section presents the inequalities used in the BAC algorithm: The generalized subtour elimination constraints (constraints (6) the mathematical model), the 0-1 knapsack cover inequalities, and the generalized capacity inequalities for the ESPPRC.

4.1 Generalized Subtour Elimination Constraints

These constraints are generalizations of the subtour elimination constraints known from the traveling salesman problem, which are also valid for ESPPRC on the form:

$$x(E(S)) \leq |S| - 1 \qquad \forall S \subset V \tag{8}$$

Restricting the constants on the right-hand side to reflect the actual node flow provides a tighter inequality, since $y_i \leq 1$ for all $i \in V \setminus \{s,t\}$. The generalized subtour elimination constraints can be written on either of the forms:

$$x(E(S)) \leq y(S) - y_i \qquad \forall i \in S, \forall S \subset V \tag{9}$$

$$x(\delta(S)) \geq 2y_i \qquad \forall i \in S, \forall S \subset V \setminus \{s,t\} \tag{10}$$

Separation of (9) and (10) can be done by solving a minimum cut problem from each node $i \in V \setminus \{s,t\}$ to the target node t (or the source node s) on the induced graph of the LP solution $(x^\star, y^\star)$ with edge weights w_e given as:

$$w_e = \begin{cases} x_e^\star & e \in E \setminus \{(s,t)\} \\ M & e = (s,t) \end{cases}$$

where M is a sufficiently large constant to ensure that s and t are on the same side of the cut, see Wolsey [30].

4.2 0-1 Knapsack Cover Inequalities

A 0-1 knapsack cover inequality for a set of nodes $S \subseteq V$ where $\sum_{i \in S} d_i^r > Q^r$ for some $r \in R$ is given as:

$$y(S) \leq |S| - 1 \tag{11}$$

The inequality states, that if a set of nodes violates the upper bound on the resource r, then not all nodes in the set can be visited by the path. The 0-1 knapsack cover inequality (11) can be rewritten as

$$\sum_{i \in S} (1 - y_i) \geq 1 \tag{12}$$

Given the LP solution (x^*, y^*), the separation problem becomes finding a cover S, i.e, a set $S \subseteq V$ satisfying $\sum_{i \in S} d_i^r > Q^r$ for some $r \in R$ such that

$$\sum_{i \in S} (1 - y_i^*) < 1 \tag{13}$$

in which case the corresponding 0-1 knapsack cover inequality (11) is violated. The most violating (11) is identified by minimizing the left-hand side of (13) for all $r \in R$, i.e., by solving:

$$\zeta = \min_{r \in R} \left\{ \min_{S \subseteq V} \left\{ \sum_{i \in S} (1 - y_i^*) z_i : \sum_{i \in S} d_i^r z_i > Q^r, z \in \{0,1\}^{|V|} \right\} \right\}$$

If $\zeta \geq 1$, no cover that violates (11) exists. The separation problem consists of $|R|$ minimization versions of the well known 0-1 knapsack problem, see Kellerer et al. [21], Wolsey [30].

Jepsen and Spoorendonk [19] suggested to exploit the fact that, since $y_i \leq 1$ for all $i \in V \setminus \{s,t,\}$, the flow through a set of nodes S can be less than 2 in an LP solution. That is, scaling the right-hand side of (11) with half the flow $x(\delta(S))$ yields

$$y(S) \leq \frac{1}{2}(|S| - 1)x(\delta(S)) \tag{14}$$

When $x(\delta(S)) < 2$, there are cases where the inequality (14) is violated and the normal 0-1 knapsack cover inequality (11) is not. Jepsen and Spoorendonk [19] suggested an enumeration scheme to separate the inequalities. Their results indicated, that (14) did improve the lower bound in the root node, but had a negative effect on the convergence of the BAC algorithm. Therefore, this family of inequalities are not pursued further in this paper.

4.3 Generalized Capacity Inequalities

This subsection introduces a family of inequalities inspired by the fractional capacity inequalities of the capacitated vehicle routing problem (CVRP), see Toth and Vigo [29]. The generalized capacity inequalities are given as:

$$\frac{1}{2}Q^r x(\delta(S)) \geq \sum_{i \in S} d_i^r y_i \qquad S \subseteq V \setminus \{s,t\}, r \in R \tag{15}$$

The inequalities ensure that a set S of nodes are visited according to their demand, e.g., if 2/3 of the resource is consumed in S, then the flow in and out of S should be at least 4/3. An example of a violated (15) can be seen in Figure 4.3.

The validity of (15) is proved in the following proposition:

Proposition 1. The generalized capacity inequalities (15) are valid for the ESPPRC.

Proof. If $y(S) = 0$ then $x(\delta(S)) = 0$, therefore both the left-hand side and the right hand side evaluate to 0. If $y(S) \geq 1$ then $x(\delta(S)) \geq 2$ and due to the resource constraint (5) for resource r, the right-hand side can never evaluate to more than Q^r which will be the minimal value of the left-hand side, i.e., in this case the resource constraint (5) for resource r dominates the generalized capacity inequality. □

Given an LP solution $(x^\star, y^\star)$ the separation problem of (15) is the problem of finding a set $S \subseteq V \setminus \{s,t\}$ for a resource $r \in R$ such that

$$\begin{aligned}
& \frac{1}{2}Q^r x^\star(\delta(S)) < \sum_{i \in S} d_i^r y_i^\star \\
\Leftrightarrow \quad & \frac{1}{2}Q^r x^\star(\delta(S)) - \sum_{i \in S} d_i^r y_i^\star + \sum_{i \in V} d_i^r < \sum_{i \in V} d_i^r \\
\Leftrightarrow \quad & \frac{1}{2}Q^r x^\star(\delta(S)) + \sum_{i \in S} d_i^r (1 - y_i^\star) + \sum_{i \in V \setminus S} d_i^r < \sum_{i \in V} d_i^r
\end{aligned}$$

Separating (15) for an be done by solving $|R|(|V|-2)$ different minimum cut problems one from each node $h \in V \setminus \{s,t\}$ to the target node t for each resource $r \in R$. The problems are solved as maxflow problems using the same procedure as for separating (9) and (10). The maxflow problem for each h is solved on a directed graph induced from the LP solution $(x^\star, y^\star)$, i.e., edges are split into opposite directed arcs, and the arcs into h are disregarded. The edge weights e_{ij} are given as:

$$w_{ij} = \begin{cases} \frac{1}{2}Q^r x_{hj}^\star + d_j^r & i = h,\ j \in V \setminus \{h,t\} \\ \frac{1}{2}Q^r x_{it}^\star + d_i^r (1 - y_i^\star) & i \in V \setminus \{s,t\},\ j = t \\ \frac{1}{2}Q^r x_{ij}^\star & i \in V \setminus \{h,t\},\ j \in V \setminus \{h,t\} \\ M & i = s, j = t \end{cases}$$

Consider the fractional solution given by the graph to the right with different fractional edge values indicated by the dotted and dashed lines. The nodes are numbered $0, \ldots, 5$ where a path is sought from node 0 to 0. For a single resource, the resource demands are given as $d = \{0, 2, 2, 2, 2, 1\}$ and the resource upper bound Q is 5.

Consider a generalized capacity inequality (15) covering the node set $S = \{1, 2, 3\}$ resulting in a fractional flow $x^\star(\delta(S)) = x^\star_{01} + x^\star_{03} = \frac{4}{3}$ through the node set. The corresponding (15) is violated since

$$\frac{1}{2} Q x^\star(\delta(S)) = \frac{10}{3} \not\geq \sum_{i \in S} d_i y^\star_i = \frac{12}{3}$$

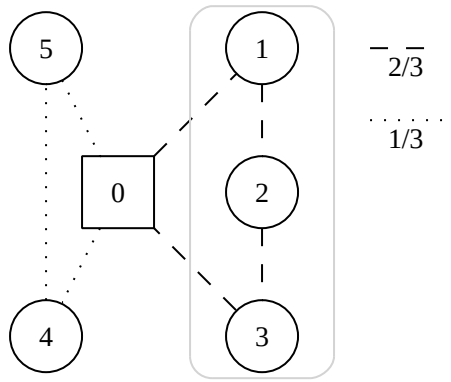


Figure 1: A violated generalized capacity inequality (15).

where M is a sufficiently large constant to ensure that s and t are on the same side of the cut. The induced graph is denser than the induced graph used for separating (9) and (10), therefore the separation of (15) is expected to be slower.

5 Computational Results

The experiments begin with an investigation of the impact of the parameter settings for the cut generation of the generalized subtour elimination constraints (9). Next, the impact of the generalized capacity inequalities (15) are investigated. For the parameter test, we consider 10 of the harder problems of the generated instances. This is followed by a lower bound comparison on the generated instances using different separation strategies. Last is a comparison of the BAC algorithm and a labeling algorithm. We use a labeling algorithm, that is implemented as described in Righini and Salani [24]. For the known instances, the comparison is made with the results obtained in Righini and Salani [25]. The mathematical model for the ESPPRC presented in this paper contains an exponential number of constraints, so it is not possible to input it directly into a general purpose mixed integer solver such as ILOG's CPLEX. However, it is possible to model the globally constrained resources in a similar way as the locally constrained resources, e.g., with the MTZ constraints. Such a model can be plugged into CPLEX and solved directly, but preliminary results indicate that this approach is always significantly slower than using the BAC algorithm proposed in this paper.

All experiments are performed on a 2.66 GHz Intel(R) Xeon(R) X5355 machine with 8 GB memory using CPLEX 10.2. The BAC algorithm is implemented using callback functions for the cut generation, which is available in the CPLEX callable library. The tests are performed using the default CPLEX parameters. This includes the generation of cuts for general mixed-integer programs such as Chvátal-Gomory, mixed-integer rounding, and disjunctive cuts.

Also, the 0-1 knapsack covers are included in the CPLEX default settings and preliminary tests indicated, that the separation time nor the change in lower bounds were much affected by the cuts. Therefore, we have not performed any further tests of the 0-1 knapsack covers but rely on the CPLEX default settings.

5.1 The Benchmark Instances

A set of benchmarks derived from the CVRP instances (divided in series A, B, E, G, M, and P) available at `http://www.branchandcut.org` has been generated. Here, the source and target nodes are chosen by splitting the node representing the depot in two. To identify sufficiently hard instances of the ESPPRC, we have used the BAC algorithm for the ESPPRC in a simple column generation algorithm for the CVRP, see e.g., Baldacci et al. [5] for the details on mathematical models. We have not included results for the CVRP, since it is not in the scope of this paper. Note, that for all the generated instances there is a valid upper bound of 0, since they are constructed from a column generation algorithm. The instances are named from the derived CVRP instances, which are given as letter indicating the series followed by the number of nodes and vehicles (the latter is not used for the ESPPRC). At the end a number, indicating the final iteration number of our column generation algorithm, is added, e.g., the instance P-n50-k7-92 is from the P-series and consists of 50 nodes (where 7 vehicles are used for the CVRP), and is from iteration 92. The ESPPRC instances are gathered in the SPPRCLIB available at `http://www.diku.dk/~spooren/spprclib.htm`.

Beside the generated instances, we consider the instances used in Feillet et al. [16], Righini and Salani [24, 25] with 100 nodes and a single globally constrained resource (the capacity resource). These instances are derived from the benchmarks by Solomon [26] for the vehicle routing problem with time windows, where the time constraints have been discarded. For the c101, r101, and rc101, three different distributions of nodes are chosen, and ten instances have been created for each distribution, where the resource bounds (capacity) range from 10 to 100 in steps of 10. We consider only instances with bounds of 60 and above. Additionally, we have extended the set of instances by setting bounds to 200, 500, 700, and 1000. A larger resource bound results in loosely constrained instances, that are expected to be harder to solve to optimality. The instances are named according to the series and a tenth of the capacity, e.g., c_100_09 is from the c101 instance, with capacity 90.

5.2 Impact of the Parameters for the Generalized Subtour Elimination Constraints

The setting of the parameters for the generation of violated generalized subtour elimination constraints (6) can have a huge influence on the computation time of the BAC algorithm. A low threshold on violation will result in good lower bounds and fewer branch nodes, but a slower convergence in each node, while the opposite is true for a high threshold. Also, the number of violated cuts added in each iteration can influence the convergence and the time spent when reoptimizing the LP-problem.

Figure 2 shows a plot with two axes given as the violation threshold and number of cuts to add per iteration. The requirement of violation is ranging from 0.1 to 1 in steps of 0.1, and the number of cuts to add is starting at 1 and then from 10 to 100 in steps of 10. The vertical axis indicates the average time spent. The time for each instance is scaled to the interval $]0, 1]$ where 1 is the maximum time given for all the parameter settings for that instance.

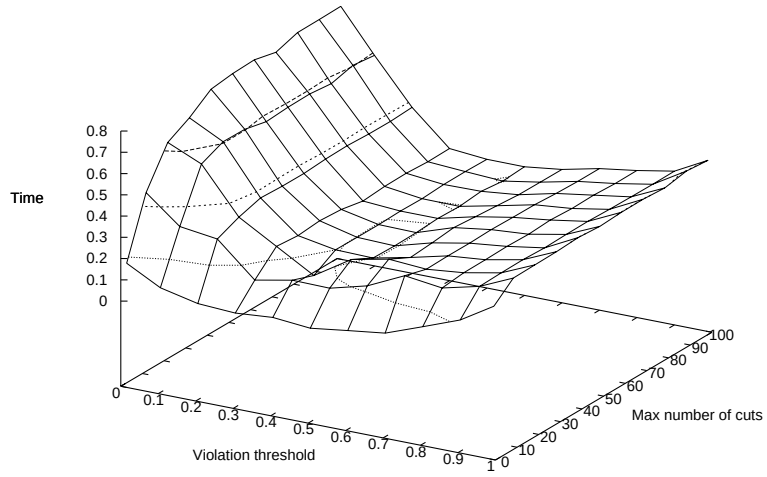


Figure 2: Parameter test for the generalized subtour elimination constraints (9). Above is a plot of the average time given the violation threshold and the number of cuts to add.

From Figure 2, it is observed that the best parameter setting appears to be to add 1 cut per iteration with a violation of at least 0.4. This indicates that the cut separation time is insignificant compared to solving the LPs.

5.3 Investigating the Generalized Capacity Inequalities

Note, that the generalized capacity inequalities (15) can substitute the generalized subtour elimination constraints (9) in the model (1)-(7), since any infeasible integer solution will be violated by some generalized capacity inequality. However, due to the computational expensive separation routine for constraints (15), a cut policy was chosen such that constraints (15) are only separated (and possible added) whenever no violated constraints (9) are separated (using the default parameters found above). Preliminary tests indicated, that due to a computational expensive separation routine for constraints (15), the cuts were not worth the effort. A slow separation was expected since the max-flow calculations are done on very dense graphs compared to the very sparse graph used in the separation of constraints (9). However, we believe that constraints (15) may become useful, e.g., with the use of a faster heuristic separation routine.

Figure 3 shows, as before, a plot of the violation threshold, number of cuts to add per iteration, and average time. The time is calculated without the separation time of constraints (15), and therefore only indicates if the convergence of the BAC is improved or not, when constraints (15) are added. Figure 3 indicates that a large violation threshold (≥ 0.8) is preferred for constraints (15) and that, the convergence of the BAC algorithm is faster when few of the constraints (15) are added. Figure 4 substantiate this result, as it can be seen that almost no cuts are added with violation thresholds 0.8 and higher. Although the generalized capacity inequalities (15) are a theoretically interesting set of inequalities, our tests have shown that in their current form and with the proposed exact separation routine, the inequalities do not appear to be computationally competitive.

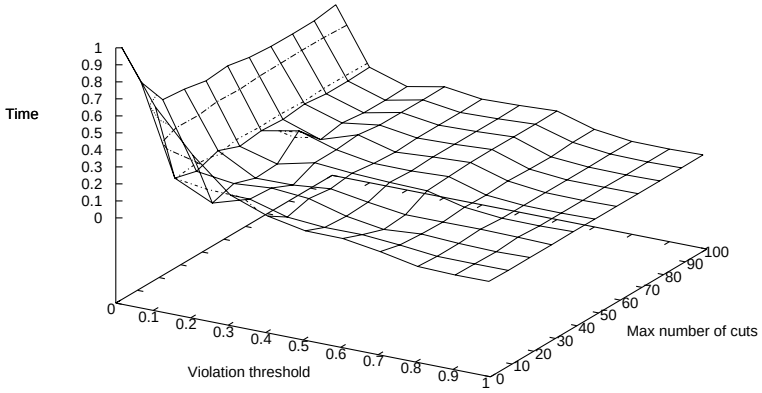


Figure 3: Parameter test for the generalized capacity inequalities (15). Above is a plot of the average time given the violation threshold and the number of cuts to add.

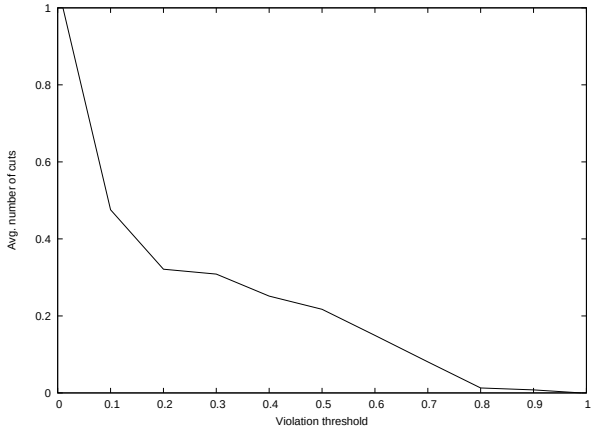


Figure 4: Parameter test for the generalized capacity inequalities (15). Above is given the average scaled number of generalized capacity inequalities added with different violation thresholds when solving the instances, i.e., with a violation threshold of 0.1 the number of cuts are decreased by about 50 % compared to the setting with a violation threshold of 0.01.

5.4 Lower Bound Comparison

Table 1 sums up the root lower bounds (root) and the number of branch nodes (nodes) for three different cut separation parameter settings. A '-' entry in the branch node columns indicates that the BAC algorithm timed out at 600 seconds. The three parameter settings tested are:

- `GSEC` is the BAC algorithm where at most 1 violated generalized subtour elimination constraint (9) with a minimum violation of 0.01 is added per iteration.
- `GCI` is the BAC algorithm with the `GSEC` parameter setting and when no violated (9) are found then at most 1 violated generalized capacity inequality (15) with a minimum violation of 0.01 is added.
- `default` is the BAC algorithm where at most 1 violated generalized subtour elimination constraint (9) with a minimum violation of 0.4 is added per iteration.

The optimal solution is given in the rightmost column.

When comparing the parameter settings `GSEC` and `GCI`, it is obvious that the generalized capacity inequalities (15) improve the lower bounds considerably. The average gap is decreased by 63% when comparing the two settings, this includes the instances that timed out and potentially could have improved the lower bound further. Surprisingly, the number of branch nodes does not decrease proportionally with the size of the gap. That is, for the instances that did not time out, the average gap is closed by 76% but with only 7% fewer branch nodes. In several cases, the number of branch nodes actually increases considerably (A-n63-k9-157, B-n45-k6-54, P-n50-k10-24, P-n55-k10-44). This indicates that (15) complicates the branch decisions. The comparison of the settings `GSEC` and `default` is more as expected: A worse lower bound with the `default` setting leads to more branch nodes. However, the previous test for the generalized subtour elimination (9) constraints showed, that this setting was the fastest on average.

5.5 Comparison with a Labeling Algorithm

Table 2 shows the running time of the BAC algorithm (BAC time (s)) with default parameters compared to the running time of our implementation of a labeling algorithm (LA) (LA time (s)) for the generated instances. The time limit was set to two hours and a timeout is indicated with a '-' in the table. The rightmost column presents the speed up if both algorithms finished. The BAC algorithm clearly outperforms the labeling algorithm. That is, in all 45 instances. However, it is worth noting that when the solution is near 0 (which is and upper bound for all instances since they are generated as pricing problems in a column generation algorithm) then the labeling algorithm performs much better than on the instances that contains much negativity. That is, the label algorithm is faster when there are less negativity in the problem whereas the BAC algorithm appears to be more robust. It should be noted that the implementation of our labeling algorithm may be improved, but it is doubtful, that it will be competitive with the BAC algorithm for the instances with a speed up of more than 100.

	GSEC		GCI		default		
Name	nodes	root	nodes	root	nodes	root	solution
A-n54-k7-149	231	-90877	-	-41213	280	-109018	-12492
A-n60-k9-57	1641	-98206	-	-64557	3071	-118437	-1000
A-n61-k9-80	205	-63534	92	-41032	462	-73397	-23549
A-n62-k8-99	133	-103839	-	-47340	301	-122973	-35969
A-n63-k9-157	122	-63082	492	-38929	113	-78190	-24189
A-n63-k10-44	127	-76475	149	-51035	280	-80765	-32561
A-n64-k9-45	358	-92812	157	-65209	425	-104686	-50550
A-n65-k9-10	152	-93117	129	-58526	189	-103936	-42835
A-n69-k9-42	72	-56453	-	-53299	179	-60410	-43290
A-n80-k10-14	84	-121510	45	-112483	120	-128508	-105283
B-n45-k6-54	277	-95588	497	-88761	502	-103214	-74278
B-n50-k8-40	166	-105497	-	-41212	237	-128488	-12832
B-n52-k7-15	25	-85997	22	-79129	59	-90278	-74998
B-n57-k7-20	12	-876421	19	-876421	328	-882924	-867154
B-n66-k9-50	239	-81006	28	-38097	1195	-94120	-26520
B-n67-k10-26	184	-55180	178	-26808	343	-63086	-21924
B-n68-k9-65	150	-88375	-	-55175	342	-99383	-31001
B-n78-k10-70	344	-91021	-	-54330	480	-101516	-44333
E-n76-k7-44	117	-30127	115	-25885	338	-32038	-22214
E-n76-k10-72	239	-36569	164	-31404	138	-38613	-25241
E-n76-k14-102	3163	-28126	-	-16153	3992	-31018	-1
E-n76-k15-40	3747	-25752	-	-17526	4993	-28675	-1
E-n101-k8-291	48	-8296	-	-7398	197	-9472	-4266
E-n101-k14-158	1468	-25748	-	-22729	1350	-30882	-3590
G-n262-k25-316	669	-1434843	-	-1434843	1510	-1434883	-1426535
M-n101-k10-97	40	-35323	37	-34758	76	-37825	-32628
M-n121-k7-260	89	-162680	-	-161424	147	-164742	-160097
M-n151-k12-15	338	-87899	-	-85488	822	-92880	-79996
M-n200-k16-143	6	-199411	4	-199411	118	-201772	-198792
M-n200-k17-12	4	-121506	1	-121210	7	-121506	-121210
P-n50-k7-92	950	-18594	1152	-12245	1319	-21516	-2
P-n50-k8-19	160	-89868	40	-89848	207	-90606	-83307
P-n50-k10-24	197	-19811	608	-11971	443	-21975	-2965
P-n51-k10-30	1028	-23812	-	-18488	1588	-27061	-2
P-n55-k7-116	84	-27065	36	-22945	105	-28094	-17824
P-n55-k8-260	101	-18839	145	-11377	167	-22237	-3573
P-n55-k10-44	913	-21448	2197	-11798	1192	-25131	-1090
P-n55-k15-88	5971	-26723	-	-20135	4781	-28128	-2
P-n60-k10-24	242	-26948	137	-21183	301	-29289	-15001
P-n60-k15-8	1495	-21889	1889	-13812	1507	-24674	-534
P-n65-k10-102	2390	-18923	-	-10975	2532	-21424	-3
P-n70-k10-12	2	-72264	1	-70317	21	-73460	-70317
P-n76-k4-41	1	-88276	1	-88276	1	-88276	-88276
P-n76-k5-16	6	-108884	10	-108884	24	-108884	-107633
P-n101-k4-174	174	-19656	165	-19041	395	-19887	-17702

Table 1: Comparison of the number of branch nodes and lower bounds for the generated instances using three different cut separation strategies.

Name	BAC time (s)	LA time (s)	speed up
A-n54-k7-149	6.96	1735.23	249.3
A-n60-k9-57	36.55	242.64	6.6
A-n61-k9-80	4.44	-	∞
A-n62-k8-99	17.94	-	∞
A-n63-k9-157	3.16	-	∞
A-n63-k10-44	2.12	693.80	327.3
A-n64-k9-45	14.57	-	∞
A-n65-k9-10	4.43	-	∞
A-n69-k9-42	1.76	3246.72	1844.7
A-n80-k10-14	12.14	-	∞
B-n45-k6-54	1.32	-	∞
B-n50-k8-40	11.01	-	∞
B-n52-k7-15	1.00	-	∞
B-n57-k7-20	1.74	-	∞
B-n66-k9-50	66.93	-	∞
B-n67-k10-26	4.62	-	∞
B-n68-k9-65	11.88	-	∞
B-n78-k10-70	24.30	-	∞
E-n76-k7-44	6.02	-	∞
E-n76-k10-72	1.19	-	∞
E-n76-k14-102	14.77	45.19	3.1
E-n76-k15-40	19.59	151.59	7.7
E-n101-k8-291	8.08	-	∞
E-n101-k14-158	37.84	-	∞
G-n262-k25-316	53.00	-	∞
M-n101-k10-97	3.12	-	∞
M-n121-k7-260	34.46	-	∞
M-n151-k12-15	78.03	-	∞
M-n200-k16-143	3.18	-	∞
M-n200-k17-12	17.75	-	∞
P-n50-k7-92	2.42	104.22	43.1
P-n50-k8-19	0.36	-	∞
P-n50-k10-24	0.72	2.91	4.0
P-n51-k10-30	2.18	4.06	1.9
P-n55-k7-116	0.58	2275.07	3922.5
P-n55-k8-260	1.20	133.45	111.2
P-n55-k10-44	2.14	14.69	6.9
P-n55-k15-88	3.97	44.73	11.3
P-n60-k10-24	1.04	110.20	106.0
P-n60-k15-8	1.95	2.50	1.3
P-n65-k10-102	6.65	163.48	24.6
P-n70-k10-12	0.24	-	∞
P-n76-k4-41	1.85	-	∞
P-n76-k5-16	0.57	-	∞
P-n101-k4-174	11.25	-	∞
Best	45	0	

Table 2: Time comparison of the BAC algorithm and the labeling algorithm.

Name	BAC time (s)	DSSR time (s)
c_100_06	0.36	0.21
c_100_07	0.38	0.18
c_100_08	0.53	1.34
c_100_09	0.62	2.02
c_100_10	1.14	7.68
c_100_20	0.82	n.a.
c_100_50	3.07	n.a.
c_100_70	2.70	n.a.
c_100_100	4.43	n.a.
r_100_06	0.75	34.64
r_100_07	0.85	143.63
r_100_08	1.35	281.62
r_100_09	1.04	1002.30
r_100_10	0.80	-
r_100_20	2.09	n.a.
r_100_50	26.96	n.a.
r_100_70	16.25	n.a.
r_100_100	1.76	n.a.
rc_100_06	0.23	0.35
rc_100_07	0.66	0.92
rc_100_08	0.90	1.77
rc_100_09	0.36	1.40
rc_100_10	0.77	7.33
rc_100_20	1.08	n.a.
rc_100_50	4.10	n.a.
rc_100_70	4.17	n.a.
rc_100_100	6.47	n.a.
Best	28 (13)	2

Table 3: Time comparison of the BAC algorithm and the labeling algorithm (Righini and Salani [25]).

In Table 3 the BAC algorithm is compared to the results obtained with the decremental state-space relaxation (DSSR) algorithm by Righini and Salani [25] (recall from Section 1 that this a specialized labeling algorithm). The running times for the two algorithms are given in the columns (BAC time (s)) and (DSSR time (s)). Since Righini and Salani [25] performed their tests on a 1.6 GHz Intel (R) Pentium 4(R) with 512 MB memory, and an exact time comparison with our machine is hard, so we have not included the speed up factor. '-' indicates that the algorithm timed out after one hour, the 'n.a.' entry indicates that no result is available for that instance.

Although the DSSR algorithm is faster on two instances out of the 15 comparable cases, it is only marginally better (even when taken their slower machine into account). There is a clear tendency, that when the capacity increases (i.e., when the ESPPRC becomes more loosely constrained) the running times of the DSSR algorithm increase significantly. The running times are also generally increasing for the BAC algorithm when the capacity increases (except for r_100_100), but not as drastically as for the DSSR algorithm. Results are not available for the DSSR algorithm for the extended instances (with capacity from 200 and above), but if the tendency from the smaller instances continues, then the DSSR algorithm will probably not be able to solve the larger instances within the time limit. The BAC algorithm is clearly superior for the loosely constrained instances.

6 Concluding Remarks

This paper introduces a BAC algorithm for solving the ESPPRC. The algorithm clearly outperformed the labeling algorithms (our own implementation of the one describes in Righini and Salani [24] as well as the one by Righini and Salani [25]) for the tested instances. Labeling algorithms have been the preferred solution approach up until now, but the experimental results presented in this paper suggest otherwise. Furthermore, the generalized capacity inequalities were introduced as a set of valid inequalities for the ESPPRC. It can be concluded that the inequalities improve the lower bounds significantly. However, this comes at a cost of complicating the branch decision, and leads to a large amount of additional branch nodes. Also, the exact separation routine takes a considerable amount of time. This is due to solving a maxflow problem on an almost complete graph. That is, the generalized capacity inequalities improve the lower bound, but lead to increased running times.

Future research could include the adaption of more valid inequalities known from related problems, e.g., two-matching inequalities, comb inequalities, and infeasible path inequalities. Another interesting direction is the conditional cuts by Fischetti et al. [17]. Such cuts resemble a specialized branch rule, as they cut off some of the branch tree after solving a subproblem that finds the optimal solution for the subtree. Another natural extension of the work presented in this paper is to extend the BAC algorithm to include locally constrained resources. This would lead to a larger mathematical formulation and will most definitely pose a serious challenge for future research.

References

[1] N. Ascheuer, M. Fischetti, and M. GrÃ¶tschel. A polyhedral study of the asymmetric travelling salesman problem with time windows. *Networks*, 36(2):69–79, 2000. doi: 10.1002/1097-0037(200009)36:2⟨69::AID-NET1⟩3.0.CO;2-Q.

[2] N. Ascheuer, M. Fischetti, and M. GrÃ¶tschel. Solving the asymmetric travelling salesman problem with time windows by branch-and-cut. *Mathematical Programming*, 90(3): 475–506, 2001. doi: 10.1007/PL00011432.

[3] E. Balas. The prize collecting traveling salesman problem. *Networks*, 19(6):621–636, 1989. doi: 10.1002/net.3230190602.

[4] E. Balas. The prize collecting traveling salesman problem: Ii. polyhedral results. *Networks*, 25(4):199–216, 1995. doi: 10.1002/net.3230250406.

[5] R. Baldacci, N. Christofides, and A. Mingozzi. An exact algorithm for the vehicle routing problem based on the set partitioning formulation with additional cuts. *Mathematical Programming*, 115(2):351–385, 2008. doi: 10.1007/s10107-007-0178-5.

[6] P. Bauer, J. T. Linderoth, and M. W. P. Savelsbergh. A branch and cut approach to the cardinality constrained circuit problem. *Mathematical Programming*, 91(2):307–348, 2002. doi: 10.1007/s101070100209.

[7] J.E. Beasley and N. Christofides. An algorithm for the resource constrained shortest path problem. *Networks*, 19:379–394, 1989. doi: 10.1002/net.3230190402.

[8] N. Boland, J. Dethridge, and I. Dumitrescu. Accelerated label setting algorithms for the elementary resource constrained shortest path problem. *Operation Research Letters*, 34 (1):58–68, 2006. doi: 10.1016/j.orl.2004.11.011.

[9] W. M. Carlyle and R. K. Wood. Near-shortest and k-shortest simple paths. *Networks*, 46(2):98–109, 2005. ISSN 0028-3045. doi: 10.1002/net.v46:2.

[10] W.M. Carlyle, J.O. Royset, and R.K. Wood. Lagrangian relaxation and enumeration for solving constrained shortest-path problems. *Networks*, 51(3):155–170, 2008. doi: 10.1002/net.20212.

[11] A. Chabrier. Vehicle routing problem with elementary shortest path based column generation. *Computers & Operations Research*, 33(10):2972–2990, 2006. doi: 10.1016/j.cor.2005.02.029.

[12] G. Desaulniers, J. Desrosiers, J. Ioachim, I. M. Solomon, F. Soumis, and D. Villeneuve. A unified framework for deterministic time constrained vehicle routing and crew scheduling problems. In T. G. Crainic and G. Laporte, editors, *Fleet Management and Logistics*, pages 57–93. Kluwer, 1998.

[13] G. Desaulniers, F. Lessard, and A. Hadjar. Tabu search, partial elementarity, and generalized k-path inequalities for the vehicle routing problem with time windows. *Transportation Science*, 42(3):387–404, 2008. doi: 10.1287/trsc.1070.0223.

[14] M. Dror. Note on the complexity of the shortest path models for column generation in VRPTW. *Operations Research*, 42(5):977–979, 1994. doi: 10.1287/opre.42.5.977.

[15] I. Dumitrescu and N. Boland. Improved preprocessing, labeling and scaling algorithms for the weight-constrained shortest path problem. *Networks*, 42(3):135–153, 2003. doi: 10.1002/net.10090.

[16] D. Feillet, P. Dejax, M. Gendreau, and C. Gueguen. An exact algorithm for the elementary shortest path problem with resource constraints: Application to some vehicle routing problems. *Networks*, 44(3):216–229, 2004. doi: 10.1002/net.v44:3.

[17] M. Fischetti, J. J. Salazar-Gonzalez, and P. Toth. Solving the orienteering problem through branch-and-cut. *INFORMS Journal on Computing*, 10(2):133–148, 1998. ISSN 1526-5528. doi: 10.1287/ijoc.10.2.133.

[18] R. Fukasawa, H. Longo, J. Lysgaard, M. Poggi de AragÃ£o, M. Reis, E. Uchoa, and R.F. Werneck. Robust branch-and-cut-and-price for the capacitated vehicle routing problem. *Mathematical Programming*, 106(3):491–511, 2006. doi: 10.1007/s10107-005-0644-x.

[19] M. Jepsen and S. Spoorendonk. A note on the flow extended 0-1 knapsack cover inequalities for the elementary shortest path problem with a capacity constraint. Technical Report 08-02, DIKU Department of Computer Science, University of Copenhagen, Denmark, 2008.

[20] M. Jepsen, B. Petersen, S. Spoorendonk, and D. Pisinger. Subset-row inequalities applied to the vehicle-routing problem with time windows. *Operations Research*, 56(2):497–511, 2008. doi: 10.1287/opre.1070.0449.

[21] H. Kellerer, U. Pferschy, and D. Pisinger. *Knapsack Problems*. Springer, Berlin, Germany, 2004.

[22] C. E. Miller, A. W. Tucker, and R. A. Zemlin. Integer programming formulation of traveling salesman problems. *Journal of the ACM*, 7(4):326–329, 1960. doi: 10.1145/321043.321046.

[23] B. Petersen, D. Pisinger, and S. Spoorendonk. Chvátal-Gomory rank-1 cuts used in a Dantzig-Wolfe decomposition of the vehicle routing problem with time windows. In B. Golden, R. Raghavan, and E. Wasil, editors, *The Vehicle Routing Problem: Latest Advances and New Challenges*, pages 397–420. Springer, 2008. doi: 10.1007/978-0-387-77778-8_18.

[24] G. Righini and M. Salani. Symmetry helps: bounded bi-directional dynamic programming for the elementary shortest path problem with resource constraints. *Discrete Optimization*, 3(3):255–273, 2006. doi: 10.1016/j.disopt.2006.05.007.

[25] G. Righini and M. Salani. New dynamic programming algorithms for the resource constrained shortest path problem. *Networks*, 51(3):155–170., 2008. doi: 10.1002/net.20212.

[26] M. M. Solomon. Algorithms for the vehicle routing and scheduling problem with time window constraints. *Operations Research*, 35(2):234–265, 1987. doi: 10.1287/opre.35.2.254.

[27] S. Spoorendonk and G. Desaulniers. Clique inequalities applied to vehicle routing problem with time windows. Submitted, 2008.

[28] S. Spoorendonk, G. Desaulniers, and J. Desrosiers. A note on cutting planes in Dantzig-Wolfe decompositions of integer programs. Submitted, 2008.

[29] P. Toth and D. Vigo, editors. *The Vehicle Routing Problem*. SIAM, 2002.

[30] L. A. Wolsey. *Integer Programming*. John Wiley & Sons, Inc., 1998.

Chapter 6

Partial Path Column Generation for the Vehicle Routing Problem

Mads Jepsen
DTU Management Engineering, Technical University of Denmark

Bjørn Petersen
DTU Management Engineering, Technical University of Denmark

David Pisinger
DTU Management Engineering, Technical University of Denmark

Abstract

This paper presents a column generation algorithm for the Capacitated Vehicle Routing Problem (CVRP) and the Vehicle Routing Problem with Time Windows (VRPTW). Traditionally, column generation models of the CVRP and VRPTW have consisted of a Set Partitioning master problem with each column representing a route. The use of Elementary routes, where no customer is visited more than once, have shown superior results for both CVRP and VRPTW. However, algorithms for solving the pricing problems do not scale well when the number of feasible routes increases. We suggest to relax the constraint that 'each column is a route' into 'each column is a part of the giant tour'; a so-called partial path, i.e., not necessarily starting and ending in the depot. This way, the length of the partial path can be bounded and a better control of the size of the solution space for the pricing problem can be obtained. It is shown that the LP-relaxed partial path formulation gives a tighter bound than the LP-relaxation of a 2-index formulation, and in some cases it is even tighter than the bound found by classical decomposition into routes.

Keywords: Vehicle Routing Problem, Column Generation, Elementary Shortest Path Problem with Resource Constraints

---

In revision.

1 Introduction

The *Capacitated Vehicle Routing Problem* (CVRP) can be described as follows: A set of customers C having a demand d_i, needs to be serviced by a number of vehicles all starting and ending at a central depot. Each customer must be visited exactly once and the capacity Q of the vehicles may not be exceeded. The objective is to service all customers traveling the least possible distance. In this paper we consider a homogeneous fleet, i.e., all vehicles are identical. The *Vehicle Routing Problem with Time Windows* (VRPTW) extends the CVRP by imposing that each customer must be visited within a given time window. We will use the term VRP to denote Vehicle Routing Problems with time and/or capacity constraints.

The standard Dantzig-Wolfe decomposition of the arc flow formulation of the VRP is to split the problem into a master problem formulated as a Set Partitioning Problem, and a pricing problem formulated as an Elementary Shortest Path Problem with Resource Constraints (ESPPRC), where capacity (and time) are the constrained resources. A restricted master problem can be solved with delayed column generation and embedded in a branch-and-bound framework to ensure integrality. Applying cutting planes either in the master or the pricing problem leads to a Branch-and-Cut-and-Price algorithm (BCP). Kohl et al. [24] implemented a successful BCP algorithm for the VRPTW by applying *sub-tour elimination* constraints and *two-path* cuts, Cook and Rich [10] generalized the *two-path* cuts to *k-path* cuts, and Fukasawa et al. [19] applied a range of valid inequalities for the CVRP based on the branch and cut algorithm of Lysgaard et al. [25]. Common for these BCP algorithms is that all applied cuts are valid inequalities for the VRPTW respectively the CVRP with regard to the *original* arc flow formulation, and have a structure which makes it possible to handle values of the dual variables in the pricing problem without increasing the complexity of the problem. Fukasawa et al. [19] refer to this as a *robust* approach in their paper. The topic of column generation and BCP algorithms has been surveyed by Barnhart et al. [4] and LÃ$\frac{1}{4}$bbecke and Desrosiers [26]. Recently the BCP framework was extended to include valid inequalities for the master problem, more specifically by applying the subset row (SR) inequalities to the Set Partitioning master problem in Jepsen et al. [23] and later by applying Chvátal-Gomory Rank-1 (CG1) inequalities in Petersen et al. [28]. Desaulniers et al. [13] solved several unsolved instances by adding generalized k-Path inequlities and generated columns heuristically using a tabu search and finally introduced a new algorithm to solve the pricing problem where partial elementarity is used. Baldacci et al. [2] improved the lower bound by adding strengthened capacity inequalities and clique inequalities to an algorithm where columns with potentially negative reduced cost are enumerated (after good upper and lower bounds are found).

Dror [16] showed that the ESPPRC, with time and capacity constraints, is strongly $\mathcal{NP}$-hard. Hence, a relaxation of the ESPPRC was used as the pricing problem in earlier BCP approaches for the VRPTW. The relaxed pricing problem where non-elementary paths are allowed is denoted the Shortest Path Problem with Resource Constraints (SPPRC) and can be solved in pseudo-polynomial time by dynamic programming using for instance a labeling algorithm, see Desrochers [14]. Considering a single capacity resource Christofides et al. [9] suggested to remove 2-cycles from the paths. This was later generalized to the variant with time windows by Desrochers et al. [15]. Irnich and Villeneuve [22] extended the framework further to k-cycle elimination (k-cyc-SPPRC), where cycles containing k or less nodes are forbidden.

Beasley and Christofides [5] proposed to solve the ESPPRC using Lagrangian relaxation.

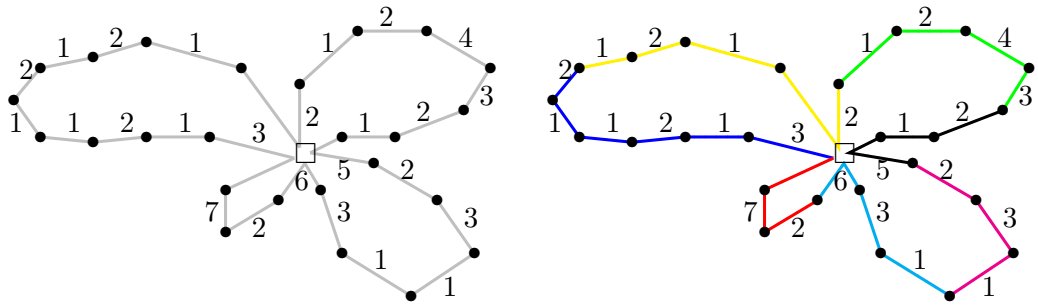


Figure 1: Giant-tour (left) and corresponding giant-tour split into partial paths (right), each bounded by the capacity $Q = 10$.

However, labeling algorithms have recently become the most popular approach to solve the ESPPRC, see e.g. Dumitrescu [17] and Feillet et al. [18]. When solving the ESPPRC with a labeling algorithm, a binary resource for each node is added, increasing the complexity of the algorithm compared to the solution of the SPPRC or the k-cyc-SPPRC. Righini and Salani [29] developed a labeling algorithm using the idea of Dijkstra's bi-directional shortest path algorithm that expands both forward and backward from the depot and connects routes in the middle, thereby potentially reducing the running time of the algorithm. Furthermore, Righini and Salani [30] and Boland et al. [6] proposed a decremental state space algorithm that iteratively solves a SPPRC, by iteratively applying binary resources to force nodes to be visited at most once. Recently Chabrier [7], Danna and Le Pape [11], and Salani [31] successfully solved several previously unsolved instances of the VRPTW from the benchmarks of Solomon [32] using a labeling algorithm for the ESPPRC. However, these algorithms have some weaknesses when dealing with very long (measured in the number of visited nodes) paths, when resource constraints are not tight. Christofides and Eilon [8] introduced the giant-tour representation in which all the routes are represented by one single *giant* tour, i.e., all the routes are concatenated into a single tour.

In this paper we propose a decomposition approach based on the generation of partial paths and the concatenation of these. The main idea is to limit the solution space of the pricing problem by bounding a resource, e.g., the number of nodes on a path or the capacity on it. The master problem combines a known number of these bounded partial paths such that all customers are visited. In this way we get a better controle of the pricing problem. If the original pricing problem is too difficult to solve for each vehicle, we may imposing a limit on the nodes in a partial path. If the original pricing problem for each vehicle is easy, we can choose looser bounds such that the partial paths get longer and lead to tighter bounds.

The paper is organized as follows: In Section 2 we describe how to use the giant tour formulation of VRP to obtain the partial path formulation. Section 3 introduces a mathematical model based on partial paths. Section 4 shows how the model is decomposed through Dantzig-Wolfe decomposition, and describes how to calculate the reduced cost of columns in a delayed column generation framework. Section 5 describes how to use the load resource to divide the solution space. Section 6 concludes the paper discussing future work.

2 Bounded Partial Paths

Given a graph $G(V, A)$ with nodes $V = C \cup \{0\}$ and arcs A, where C is the set of customers, and 0 is the depot. Moreover, we have a set R of resources which e.g. can be load and/or time. Each resource $r \in R$ has a resource window $[a_i^r, b_i^r]$ that must be met upon arrival to node $i \in V$, and a consumption $\tau_{ij}^r \geq 0$ for using arc $(i, j) \in A$. A resource consumption at a node $i \in C$ is modeled by a resource consumption at edge (i, j), and hence usually $\tau_{0j}^r = 0$ for all $j \in C$. A global capacity limit Q can be modeled by imposing a resource window $[0, Q]$ for the depot node 0.

The VRP can now be stated as: Find a set of routes starting and ending at the depot node 0 satisfying all resource windows, such that the cost is minimized and all customers C are visited.

A solution to the VRP will consist of a number of routes

$$\begin{aligned} &0 \to i_1^1 \to \ldots \to i_{k_1}^1 \to 0, \\ &0 \to i_1^2 \to \ldots \to i_{k_2}^2 \to 0, \\ &\vdots \\ &0 \to i_1^n \to \ldots \to i_{k_n}^n \to 0 \end{aligned}$$

where n is the number of vehicles, and k_j is the length of the j'th route. A natural decomposition of the VRP is to split the problem into these separate routes, where a master problem ensures that all customers are visited once. We will call this the *classical* decomposition. However, using the classical decomposition, the number of nodes in each individual route may vary a lot, making it difficult to solve some of the subproblems.

Instead we consider the giant-tour representation by Christofides and Eilon [8]

$$0 \to i_1^1 \to \ldots \to i_{k_1}^1 \to 0 \to i_1^2 \to \ldots \to i_{k_2}^2 \to 0 \to \ldots \to 0 \to i_1^n \to \ldots \to i_{k_n}^n \to 0$$

A giant-tour (see Figure 1) is one long path visiting all customers once and the depot several times. The consumption of resources $r \in R$ is reset each time the depot node is encountered. If we decompose the VRP into smaller segments of the giant-tour, we may to a larger extent controle that the number of nodes visited in each partial path is of similar length. In this way we can balance the hardness of the subproblems (see Figure 1 for an illustration).

The decompostion is done by imposing an upper limit on a resource $r' \in R$, e.g., bounding the path length in the number of nodes for each partial path, or bounding the load. The giant tour introduced in Figure 1 can be decomposed into a number of partial paths by bounding a resource. In the following the number of visited customers in C is considered to be the bounding resource. Bounding the load resource is a bit more complicated and will be addressed in Section 5.

Each segment represents a partial path of the giant-tour. With a bounded number of customers L on each partial path, K partial paths are needed to ensure that all customers are visited i.e., $L \cdot K \geq |C|$. The partial paths can start and end in any node in V and it can visit the depot several times. A partial path could for example be:

$$i_1 \to i_2 \to 0 \to i_3 \to 0 \to i_4$$

In the following we will make a graph representation for the problem of finding the K partial path of length at most L. This is done by replicating the graph K times and connecting

the replications by special arcs. Each of the replications is connected with arcs directed from one replication to a following replication. This leads to a layered graph with K layers $1, .., K$ where there are no outgoing arcs of the final layer. Each layer $k \neq K$ is connected to the subsequent layer $k+1$. Each pair of subsequent layers are connected with the set of arcs leaving node i in layer $k \neq K$ and entering layer $k+1$.

Consider the graph $G'(V', A')$ consisting of a set of layers $\mathcal{K} = \{1, \ldots, K\}$, each layer representing G for a partial path. Let G^k be the sub graph of G' representing layer k with node set $V^k = \{(i,k) : i \in V\}$ for all $k \in \mathcal{K}$ and arc set $A^k = \{(i,j,k) : (i,j) \in A\}$ for all $k \in \mathcal{K}$. Let $A^* = \{(i,i,k) : (i,k) \in V^k \wedge (i,k+1) \in V^{k+1} \wedge k \in \mathcal{K}\}$ be the set of interconnecting arcs, i.e., the arcs connecting a layer k with the layer above k namely layer $k+1$ for all $k \in \mathcal{K}$ and all nodes $i \in V$ (layer $K+1$ is defined to be layer $1 \in \mathcal{K}$ and layer 0 is defined to be layer $K \in \mathcal{K}$). Let $V' = \bigcup_{k \in \mathcal{K}} V^k$ and let $A' = \bigcup_{k \in \mathcal{K}} A^k \cup A^*$. An illustration of G' can be seen in Figure 2. Note, that arcs (i,i,k) are not present in A^k and that arcs (i,j,k) with $i \neq j$ are present in A^*, so all arcs $(i,j,k) \in A'$ can be uniquely indexed.

The resource consumption τ^r_{ij} of arcs $(i,j) \in A^k$ is the same as in the original graph A, hence we omit the index k. The resource consumption of interconnecting arcs $(i,j) \in A^*$ is $\tau^r_{ij} = 0$.

Let L be the upper bound on the length of each partial path, and let $|C|$ be the length of the combined path (the giant-tour). Now, exactly $K = \lceil |C|/L \rceil$ partial paths are needed to form the combined path, since $L \lceil |C|/L \rceil \geq |C| > L(\lceil |C|/L \rceil - 1)$. Once K has been calculated, we can further reduce the path length to $L = \lceil |C|/K \rceil$.

With the length of a path defined as the number of customers on it, the problem is now to find partial paths of length at most L in K layers with $L \cdot K \geq |C| > L \cdot (K-1)$, so that each partial path p ending in node $i \in V$ is met by another partial path p' starting in i. All partial paths are combined while not visiting any customers more than once and satisfying all resource windows. A customer $i \in C$ is considered to be on a partial path p if i is visited on p and is not the end node of p.

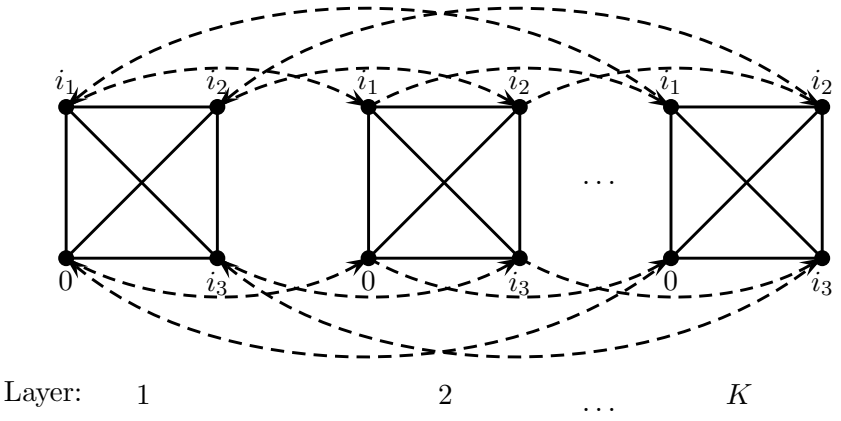


Figure 2: Illustration of G' with $|C| = 3$, $K = 3$, and $L = 1$. Full-drawn lines represent two arcs; one in each direction. Dashed lines are the interconnecting arcs A^*.

3 The Vehicle Routing Problem

We present two models for the VRP problem defined in previous section. The 2-index model is most compact, while the 3-index model is better suited for decomposition.

2-index formulation of the VRP In the following let c_{ij} be the cost of arc $(i,j) \in A$, x_{ij} be the binary variable indicating the use of arc $(i,j) \in A$, and T_{ij}^r (the resource stamp) be the consumption of resource $r \in R$ at the beginning of arc $(i,j) \in A$. Let $\delta^+(i)$ and $\delta^-(i)$ be the set of outgoing respectively ingoing arcs of node $i \in V$. Combining the two index model from Bard et al. [3] with the constraints ensuring the time windows for the ATSP by Ascheuer et al. [1] a mathematical model can be formulated as follows:

$$\min \sum_{(i,j)\in A} c_{ij}x_{ij} \tag{1}$$

$$\text{s.t.} \sum_{(i,j)\in\delta^+(i)} x_{ij} = 1 \qquad \forall i \in C \tag{2}$$

$$\sum_{(j,i)\in\delta^-(i)} x_{ji} = \sum_{(i,j)\in\delta^+(i)} x_{ij} \qquad \forall i \in V \tag{3}$$

$$\sum_{(j,i)\in\delta^-(i)} (T_{ji}^r + \tau_{ji}^r x_{ji}) \leq \sum_{(i,j)\in\delta^+(i)} T_{ij}^r \qquad \forall r \in R,\ \forall i \in C \tag{4}$$

$$a_i^r x_{ij} \leq T_{ij}^r \leq b_i^r x_{ij} \qquad \forall r \in R,\ \forall (i,j) \in A \tag{5}$$

$$T_{ij}^r \geq 0 \qquad \forall r \in R,\ \forall (i,j) \in A \tag{6}$$

$$x_{ij} \in \{0,1\} \qquad \forall (i,j) \in A \tag{7}$$

The objective (1) sums up the cost of the used arcs. Constraints (2) ensure that each customer is visited exactly once, and (3) are the flow conservation constraints. Constraints (4) and (5) ensure the resource windows are satisfied. It is assumed that the bounds on the depot are always satisfied. Note, that no sub-tours can be present since only one resource stamp per arc exists and the arc weights are positive for all $(i,j) \in A : i \in C$.

For a one dimensional resource such as *load* a stronger lower bound of the LP relaxation can be obtained by replacing (4) to (6) with $\sum_{(i,j)\in\delta^+(S)} x_{ij} \geq r(S)$, where $r(S)$ is a minimum number of vehicles needed to service the set S. All though this model can not be directly solved it is possible to overcome this problem by only including the constraints that are violated. For more details on how to separate the constraint and calculate the value of $r(S)$ the reader is refered to Toth and Vigo [33].

3-index formulation of the VRP Let x_{ij}^k be the variable indicating the use of arc $(i,j,k) \in A'$. Problem (1)–(7) is rewritten to:

$$\min \sum_{k\in\mathcal{K}} \sum_{(i,j)\in A} c_{ij} x_{ij}^k \tag{8}$$

$$\text{s.t.} \sum_{k\in\mathcal{K}} \sum_{(i,j)\in\delta^+(i)} x_{ij}^k = 1 \qquad \forall i \in C \tag{9}$$

$$\sum_{(i,j)\in\delta^+(i)} x_{ij}^k \leq 1 \qquad \forall k \in \mathcal{K},\ \forall i \in C \tag{10}$$

$$\sum_{k\in\mathcal{K}} \left(x_{ii}^{k-1} + \sum_{(j,i)\in\delta^-(i)} x_{ji}^k \right) = \sum_{k\in\mathcal{K}} \left(x_{ii}^k + \sum_{(i,j)\in\delta^+(i)} x_{ij}^k \right) \qquad \forall i \in V \tag{11}$$

$$x_{ii}^{k-1} + \sum_{(j,i)\in\delta^-(i)} x_{ji}^k = x_{ii}^k + \sum_{(i,j)\in\delta^+(i)} x_{ij}^k \qquad \forall k \in \mathcal{K},\ \forall i \in V \tag{12}$$

$$\sum_{k\in\mathcal{K}} \sum_{i\in V} x_{ii}^k = K \tag{13}$$

$$\sum_{i\in C} \sum_{(i,j)\in A} x_{ij}^k \leq L \qquad \forall k \in \mathcal{K} \tag{14}$$

$$\sum_{k\in\mathcal{K}} \sum_{(j,i)\in\delta^-(i)} \left(T_{ji}^{rk} + \tau_{ji}^r x_{ji}^k \right) \leq \sum_{k\in\mathcal{K}} \sum_{(i,j)\in\delta^+(i)} T_{ij}^{rk} \qquad \forall r \in R,\ \forall i \in C \tag{15}$$

$$\sum_{(j,i)\in\delta^-(i)} \left(T_{ji}^{rk} + \tau_{ji}^r x_{ji}^k \right) \leq \sum_{(i,j)\in\delta^+(i)} T_{ij}^{rk} \qquad \forall r \in R,\ \forall k \in \mathcal{K},\ \forall i \in C \tag{16}$$

$$a_i^r \sum_{k\in\mathcal{K}} x_{ij}^k \leq \sum_{k\in\mathcal{K}} T_{ij}^{rk} \leq b_i^r \sum_{k\in\mathcal{K}} x_{ij}^k \qquad \forall r \in R,\ \forall (i,j) \in A \tag{17}$$

$$a_i^r x_{ij}^k \leq T_{ij}^{rk} \leq b_i^r x_{ij}^k \qquad \forall r \in R,\ \forall k \in \mathcal{K},\ \forall (i,j) \in A \tag{18}$$

$$T_{ij}^{rk} \geq 0 \qquad \forall r \in R,\ \forall k \in \mathcal{K},\ \forall (i,j) \in A \tag{19}$$

$$x_{ij}^k \in \{0,1\} \qquad \forall k \in \mathcal{K},\ \forall (i,j) \in A \tag{20}$$

The objective (8) sums up the cost of the used arcs. Constraints (9) ensure that all customers are visited exactly once, while the redundant constraints (10) ensure that no customer is visited more than once. Constraints (11) maintain flow conservation between the original nodes V, and can be rewritten as

$$\sum_{k\in\mathcal{K}} \sum_{(j,i)\in\delta^-(i)} x_{ji}^k = \sum_{k\in\mathcal{K}} \sum_{(i,j)\in\delta^+(i)} x_{ij}^k \qquad \forall i \in V$$

since $\sum_{k\in\mathcal{K}} x_{ii}^{k-1} = \sum_{k\in\mathcal{K}} x_{ii}^k$. Constraints (12) maintain flow conservation within a layer. Constraint (13) ensures that K partial paths are selected and constraints (14) that the length of the partial path in each layer is at most L. Constraints (15) connect the resource variables on a global level and constraints (16) connect the resource variables within each single layer. Note, that since constraints (15) and (16) are omitted for the depot, it is not constrained by resources. Constraints (17) globally enforce the resource windows and the redundant constraints (18) enforce the resource windows within each layer.

4 Dantzig-Wolfe Decomposition

We use Dantzig-Wolfe decompostion of the 3-index formulation of the VRP, defined in (8)–(20) to reach the following master and a pricing problem. In the process of the decomposition the K identical pricing problems are combined into a single pricing problem.

4.1 Master Problem

Let λ_p a binary variable indicating whether partial path p is used. We use Dantzig-Wolfe decomposition where the constraints (9), (11), (13), (15), and (17) are kept in the master problem. Since the vehicles are identical, we can aggregate over the sets A^k getting the following master problem (PP):

$$\min \sum_{p \in P} c_p \lambda_p \tag{21}$$

$$\text{s.t.} \sum_{p \in P} \sum_{(i,j) \in \delta^+(i)} \alpha_{ij}^p \lambda_p = 1 \qquad \forall i \in C \tag{22}$$

$$\sum_{p \in P: e^p = i} \lambda_p = \sum_{p \in P: s^p = i} \lambda_p \qquad \forall i \in V \tag{23}$$

$$\sum_{p \in P} \lambda_p = K \tag{24}$$

$$\sum_{(j,i) \in \delta^-(i)} \left(T_{ji}^r + \sum_{p \in P} \tau_{ji}^r \alpha_{ji}^p \lambda_p \right) \leq \sum_{(i,j) \in \delta^+(i)} T_{ij}^r \qquad \forall r \in R,\ \forall i \in C \tag{25}$$

$$a_i^r \sum_{p \in P} \alpha_{ij}^p \lambda_p \leq T_{ij}^r \leq b_i^r \sum_{p \in P} \alpha_{ij}^p \lambda_p \qquad \forall r \in R,\ \forall (i,j) \in A \tag{26}$$

$$T_{ij}^r \geq 0 \qquad \forall r \in R,\ \forall (i,j) \in A \tag{27}$$

$$\lambda_p \in \{0,1\} \qquad \forall p \in P \tag{28}$$

In this formulation, α_{ij}^p is the number of times arc $(i,j) \in A$ is used on path $p \in P$ and s^p and e^p indicate the start respectively the end node of partial path $p \in P$. Constraints (22) ensure that each customer is visited exactly once. Constraints (23) link the partial paths together by flow conservation. Constraint (24) is the convexity constraint ensuring that K partial paths are selected. Constraints (25) and (26) enforce the resource windows.

Tightness of bounds: Before we turn our attention to the pricing problem we prove the following theorems about the quality of the bounds obtained by the decomposition.

Theorem 1. Let z_{LP} be an LP-solution to (1)–(7) and let z_{PP} be an LP-solution to (21)–(28) then $z_{LP} \leq z_{PP}$ for all instances of VRP.

Proof. $z_{LP} \leq z_{PP}$ since all solutions to (21)–(28) map to solutions to (1)–(7), see Nemhauser and Wolsey [27]. □

Theorem 2. Let z_{PP} as before be an LP-solution to (21)–(28), and z_{EP} be the LP-solution to the classical decomposition of VRP into an elementary route for each vehicle. Then instances exist where $z_{PP} > z_{EP}$.

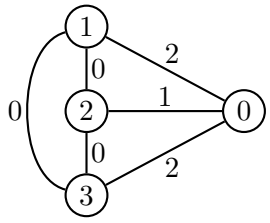


Figure 3: Three customers with demand of 1 and vehicle capacity $Q = 2$. Distances are indicated on the edges. There are six feasible routes ($\{0,1,0\}$, $\{0,2,0\}$, $\{0,3,0\}$, $\{0,1,2,0\}$, $\{0,1,3,0\}$, $\{0,2,3,0\}$) having the costs $(4,2,4,3,4,3)$. The LP solution is $(0,0,0,\frac{1}{2},\frac{1}{2},\frac{1}{2})$ with objective $z_{EP} = 5$. Using the partial path formulation with max path length $L = 3$ and $K = 1$ we find the optimal solution ($\{0,1,3,0,2,0\}$) with objective $z_{PP} = 6$.

Proof. An instance with $z_{PP} > z_{EP}$ can be constructed with three customers each with a demand of 1 and vehicle capacity $Q = 2$. Using a max path length of $L = 3$, we find $z_{PP} = 6$ while $z_{EP} = 5$. (See Figure 3). □

4.2 Pricing Problem

The K pricing problems corresponding to the master problem (21)–(28) are defined by constraints (10), (12), (14), (16), and (18) and can be formulated as a single ESPPRC where the depot is allowed to be visited more than once. Let s and e be a super source respectively a super target node. Arcs (s,i) and (i,e) for all $i \in V$ are added to G with cost and resource consumption 0.

$$\min \sum_{(i,j)\in A} \overline{c}_{ij} x_{ij} \tag{29}$$

$$\text{s.t.} \sum_{(s,i)\in\delta^+(s)} x_{si} = 1 \tag{30}$$

$$\sum_{(i,e)\in\delta^-(e)} x_{ie} = 1 \tag{31}$$

$$\sum_{(i,j)\in A} x_{ij} \leq 1 \qquad \forall i \in C \tag{32}$$

$$\sum_{(j,i)\in\delta^-(i)} x_{ji} = \sum_{(i,j)\in\delta^+(i)} x_{ij} \qquad \forall i \in V \tag{33}$$

$$\sum_{(i,j)\in A} \tau^{r'}_{ij} x_{ij} \leq L \tag{34}$$

$$\sum_{(j,i)\in\delta^-(i)} (T^r_{ji} + \tau^r_{ji} x_{ji}) \leq \sum_{(i,j)\in\delta^+(i)} T^r_{ij} \qquad \forall r \in R,\ \forall i \in C \tag{35}$$

$$a^r_i x_{ij} \leq T^r_{ij} \leq b^r_i x_{ij} \qquad \forall r \in R,\ \forall (i,j) \in A \tag{36}$$

$$x_{ij} \in \{0,1\} \qquad \forall (i,j) \in A \tag{37}$$

The objective (29) minimizes the reduced cost of a column in (PP). Constraints (30) and (31) ensure that the path starts in s respectively ends in e. Constraints (32) dictates that no node

is visited more than once, thereby ensuring elementarity. Constraints (33) conserve the flow. Constraint (34) ensures that the partial path does not use more than the allowed amount L of the restricted resource r'. Constraints (35) and (36) ensure the resource windows are satisfied for all customers. Note, since constraints (35) hold for $i \in U$ (excluding the depot), a resource is only restricted by its lower limit a_0^r for all $r \in R$ each time a path leaves the depot.

Let π ($\pi_i \geq 0 : \forall i \in C$) be the duals of (22) and $\pi_0 = 0$, let μ be the duals of (23), let $\beta \leq 0$ be the dual of (24), let ν ($\nu \leq 0 : \forall i \in C$) be the duals of (25) and $\nu_0 = 0$, and let $\underline{\omega} \leq 0$ and $\overline{\omega} \geq 0$ be the dual of (26). The cost of the arcs in this ESPPRC are then given as:

$$\overline{c}_{ij} = -\beta + \begin{cases} c_{ij} - \pi_i - \tau_{ij}\nu_j - \sum_{r \in R} a_i^r \underline{\omega}_i^r + \sum_{r \in R} b_i^r \overline{\omega}_i^r & \forall (i,j) \in A \setminus (\delta^+(s) \cup \delta^-(e)) \\ \mu_j & \forall (s,j) \in \delta^+(s) \\ -\mu_i & \forall (i,e) \in \delta^-(e) \end{cases}$$

The pricing problem is now an to find an elementary shortest path from s to e.

Solving the pricing problem: ESPPRCs can be solved by various labeling algorithms, see e.g. Desaulniers et al. [12], Irnich [20], Irnich and Desaulniers [21], and Righini and Salani [29].

Branching: Integrality can be obtained by branching on the original variables, which can be accomplished by cuts in the master problem (see Vanderbeck [34]), e.g., let X_{ij} be the set of partial paths that utilize arc (i,j) then the branch rule $x_{ij} = 0 \ \vee \ x_{ij} = 1$ can be expressed by the dichotomy:

$$\sum_{p \in X_{ij}} \lambda_p = 0 \ \vee \ \sum_{p \in X_{ij}} \lambda_p = 1.$$

5 Bounding the Load Resource

The giant tour introduced in Section 1 can be decomposed into a number of partial paths by bounding a resource r', e.g. the number of nodes, the time, or the load. In this section we consider the latter. The load constraint is present in CVRP and VRPTW and is a special type of resource constraints. If Q is the maximal load of a vehicle and $d_i : i \in C$ is the demand of the costumers, then the accumulated demand on a route may not exceed Q. The goal is that equation (34) is expressed on the form:

$$\sum_{(i,j) \in A} d_i x_{ij} \leq L$$

where L is a given threshold value for the load resource. This will potentially lead to an easier pricing problem. For dynamic programming based algorithms the complexity is dependent on the size of L. In the length case we rounded up the expression $|C|/K$ to ensure feasibility. In the following we will discuss a similar approach for bounding on the load resource.

Let the total demand of the customers be $D = \sum_{i \in C} d_i$. A lower bound on the number of partial paths needed is: $K = \lceil D/L \rceil$. However, we cannot just split the giant tour into K partial paths of capacity L since there is no guaranty that the optimal giant tour can be split into partial paths of equal capacity.

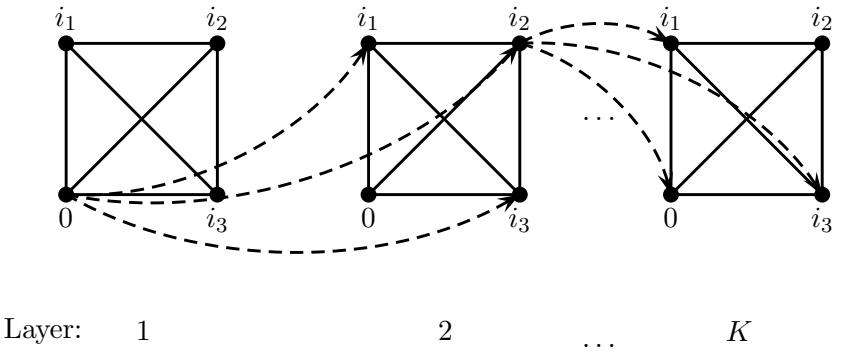


Figure 4: Small subset of the connector arcs. Connector arcs from node 0 in layer 1 to nodes in layer 2, and connector arcs from node 2 in layer 2 are shown as dashed lines. Not all connector arcs are shown due to readability of the graph.

Let the largest demand be defined as $d_{\max} = \max_{i \in C} d_i$, and assume that $L \geq d_{\max}$. Then, we need to allow up to $d_{\max} - 1$ extra capacity in each partial path, to compensate for possibly uneven splitting. This means that for a given K we find $L_{ub} = \lceil D/K \rceil + (d_{\max} - 1)$ as the upper bound on the resource consumption.

An alternative approach to increasing L to L_{ub} is to allow an additional edge exceeding L to be selected in the pricing problem. This may complicate the pricing problem, though.

The remainder of this section addresses alternative strategies to avoid complicating the pricing problem. One such alternative is to introduce the concept of connector arcs. A connector arc is a single arc between two nodes which combines two partial paths. For each layer $k \in \mathcal{K}$ and original arc $(i, j) \in A$ there is connector arc to the subsequent layer.

Figure 4 illustrates the idea of the connector arcs. The dashed lines from node 0 in layer 1 orientated towards layer 2 to node i_1, i_2 and i_3, illustrates the connectors out of node 0 in layer 1. Similar nodes i_1 in layer one will have connectors to nodes 0,i_2,i_3 in layer 2, and likewise for nodes i_2 and i_3 in layer 1 has connectors to layer 2. In layer 2 the dashed lines from node i_2 illustrates its connectors to layer 3. Similare all other nodes in layer 2 has connectors to layer 3. In layer 3 the dashed lines illustrates the final set of connectors, which are the last edges that can be used in the system and they therefor point to the depot from all nodes. The connector arcs plays the same role as the additional arc in the pricing problem suggested above. They make it possible to obtain a path which exceeds $L - 1$ by the demand of a single customer. By allowing K connector arcs it is therefore possible to obtain a solution to the problem where all the K layers include one additional node.

To model the connector arcs we introduce new variables y_{ij}^k for all $(i, j) \in A$ and for all $k \in \mathcal{K}$. These variables substitute the variables x_{ii}^k by connecting every node $(i, k) \in V^k$ in each layer $k \in \mathcal{K}$ with the nodes $(j, k+1) \in V^{k+1} : (i, j) \in A$ in the subsequent layer. Furthermore, constraints (11) are modified to:

$$\sum_{k \in \mathcal{K}} \sum_{(j,i) \in \delta^-(i)} \left(x_{ji}^k + y_{ji}^k \right) = \sum_{(i,j) \in \delta^+(i)} \left(x_{ij}^k + y_{ij}^k \right), \quad \forall i \in V$$

This ensures the global flow by taking the flow of the connector arcs into account. A similar

substitution is made in constraint (12) and (13). The connector arcs are also present in the resource constraints where they are added to any sum bounding the resource variables. Constraint (15) is therefore changed to:

$$\sum_{k\in\mathcal{K}}\sum_{(j,i)\in\delta^-(i)}\left(T_{ji}^{rk}+\tau_{ji}^{r}\left(x_{ji}^{k}+y_{ji}^{k}\right)\right)\leq\sum_{k\in\mathcal{K}}\sum_{(i,j)\in\delta^+(i)}T_{ij}^{rk},\quad\forall r\in R,\forall i\in C$$

A similar addition is made for constraints (16), (17), and (18).

When the model is decomposed into the K pricing problems each set of up to K connector arcs $y_{ij}:y_{ij}^{k},(i,j)\in A,k\in\mathcal{K}$ becomes a single connector arc connecting the paths ending in node i with the path starting in node j. Using the aggregated connector arcs constraints (23) are substituted with:

$$\sum_{p\in P:e^p=i}\lambda_p+\sum_{j\in\delta^-(i)}y_{ji}=\sum_{j\in\delta^+(i)}y_{ij}+\sum_{p\in P:s^p=i}\lambda_p\quad\forall i\in V$$

6 Conclusion and Future Work

A new decomposition model of the VRP has been presented with the ESPPRC as the pricing problem. The model makes it possible to balance the running time of the pricing problem against the tightness of the lower bound. Due to the aggregation of the model, LP relaxed bounds of (21)–(28) are better than the direct model (1)–(7). Since (21)–(28) is a generalization of the traditional Dantzig-Wolfe decomposition model with elementary routes as columns, the LP relaxed bounds may be both weaker and stronger. It has been shown that the bound of the presented LP relaxation is sometimes better than that of the classical decomposition of VRP into an elementary route for each vehicle.

Future work: The quality of the bounds can be further improved by using special purpose cutting planes, which this paper has not focused on. Furthermore, effective cuts such as Subset Row-inequalities by Jepsen et al. [23] and Chvátal-Gomory Rank-1 cuts (see Petersen et al. [28]) can be applied to the Set Partition master problem to strengthen the bound.

More and better cuts have been added to the VRPTW Branch-and-Cut algorithm used in this paper for comparison, but all of these cuts could also be added to this model obtaining at least as good a bound.

Considering the approach of Baldacci et al. [2] where columns are enumerated dependent on strong upper and lower bounds, it should be clear that the partial path approach should contain fewer enumerated columns due to the smaller solution space of the pricing problem. Combining the relatively strong bound with the small solution space a powerful strategy should be obtained.

References

[1] N. Ascheuer, M. Fischetti, and M. GrÃ¶tschel. Solving the asymmetric travelling salesman problem with time windows by branch-and-cut. *Mathematical Programming*, 90(3): 475–506, 2001. doi: 10.1007/PL00011432.

[2] R. Baldacci, N. Christofides, and A. Mingozzi. An exact algorithm for the vehicle routing problem based on the set partitioning formulation with additional cuts. *Mathematical Programming*, 115(2):351–385, 2008. doi: 10.1007/s10107-007-0178-5.

[3] J. F. Bard, G. Kontoravdis, and G. Yu. A branch-and-cut procedure for the vehicle routing problem with time windows. *Transportation Science*, 36(2):250–269, May 2002. doi: http://dx.doi.org/10.1287/trsc.36.2.250.565.

[4] C. Barnhart, E. L. Johnson, G. L. Nemhauser, M. W. P. Savelsbergh, and P. H. Vance. Branch-and-price: Column generation for solving huge integer programs. *Operations Research*, 46(3):316–329, 1998. doi: 10.1287/opre.46.3.316.

[5] J.E. Beasley and N. Christofides. An algorithm for the resource constrained shortest path problem. *Networks*, 19:379–394, 1989. doi: 10.1002/net.3230190402.

[6] N. Boland, J. Dethridge, and I. Dumitrescu. Accelerated label setting algorithms for the elementary resource constrained shortest path problem. *Operation Research Letters*, 34 (1):58–68, 2006. doi: 10.1016/j.orl.2004.11.011.

[7] A. Chabrier. Vehicle routing problem with elementary shortest path based column generation. *Computers & Operations Research*, 33(10):2972–2990, 2006. doi: 10.1016/j.cor.2005.02.029.

[8] N. Christofides and S. Eilon. An algorithm for the vehicle-dispatching problem. *Operational Research Quarterly*, 20(3):309–318, Sep 1969.

[9] N. Christofides, A. Mingozzi, and P. Toth. Exact algorithms for the vehicle routing problem, based on spanning tree and shortest path relaxations. *Mathematical Programming*, 20(1):255–282, Dec 1981. doi: 10.1007/BF01589353.

[10] W. Cook and J. L. Rich. A parallel cutting plane algorithm for the vehicle routing problem with time windows. Technical Report TR99-04, Computational and Applied Mathematics, Rice University, Houston, Texas, USA, 1999.

[11] E. Danna and C. Le Pape. Branch-and-price heuristics: A case study on the vehicle routing problem with time windows. In G. Desaulniers, J. Desrosiers, and M. M. Solomon, editors, *Column Generation*, chapter 4, pages 99–129. Springer, 2005. doi: 10.1007/0-387-25486-2_4.

[12] G. Desaulniers, J. Desrosiers, J. Ioachim, I. M. Solomon, F. Soumis, and D. Villeneuve. A unified framework for deterministic time constrained vehicle routing and crew scheduling problems. In T. G. Crainic and G. Laporte, editors, *Fleet Management and Logistics*, pages 57–93. Kluwer, 1998.

[13] G. Desaulniers, F. Lessard, and A. Hadjar. Tabu search, partial elementarity, and generalized k-path inequalities for the vehicle routing problem with time windows. *Transportation Science*, 42(3):387–404, 2008. doi: 10.1287/trsc.1070.0223.

[14] M. Desrochers. *La fabrication d2019horaires de travail pour les conducteurs d2019autobus par une mÃ©thode de gÃ©nÃ©ration de colonnes.* PhD thesis, UniversitÃ© de MontrÃ©al, MontrÃ©al, Canada, 1986.

[15] M. Desrochers, J. Desrosiers, and M. Solomon. A new optimization algorithm for the vehicle routing problem with time windows. *Operations Research*, 40(2):342–354, 1992. doi: 10.1287/opre.40.2.342.

[16] M. Dror. Note on the complexity of the shortest path models for column generation in VRPTW. *Operations Research*, 42(5):977–979, 1994. doi: 10.1287/opre.42.5.977.

[17] I. Dumitrescu. *Constrained Path and Cycle Problems*. PhD thesis, Department of Mathematics and Statistics, University of Melbourne, Australia, 2002.

[18] D. Feillet, P. Dejax, M. Gendreau, and C. Gueguen. An exact algorithm for the elementary shortest path problem with resource constraints: Application to some vehicle routing problems. *Networks*, 44(3):216–229, 2004. doi: 10.1002/net.v44:3.

[19] R. Fukasawa, H. Longo, J. Lysgaard, M. Poggi de AragÃ£o, M. Reis, E. Uchoa, and R.F. Werneck. Robust branch-and-cut-and-price for the capacitated vehicle routing problem. *Mathematical Programming*, 106(3):491–511, 2006. doi: 10.1007/s10107-005-0644-x.

[20] S. Irnich. Resource extension functions: Properties, inversion, and generalization to segments. *OR Spectrum*, 30(1):113–148, 2008. doi: 10.1007/s00291-007-0083-6.

[21] S. Irnich and G. Desaulniers. Shortest path problems with resource constraints. In G. Desaulniers, Jacques Desrosiers, and M.M. Solomon, editors, *Column Generation*, chapter 2, pages 33–65. Springer, 2005. doi: 10.1007/0-387-25486-2_2.

[22] S. Irnich and D. Villeneuve. The shortest path problem with resource constraints and k-cycle elimination for $k \geq 3$. *INFORMS Journal on Computing*, 18:391–406, 2006. doi: 10.1287/ijoc.1040.0117.

[23] M. Jepsen, B. Petersen, S. Spoorendonk, and D. Pisinger. Subset-row inequalities applied to the vehicle-routing problem with time windows. *Operations Research*, 56(2):497–511, 2008. doi: 10.1287/opre.1070.0449.

[24] N. Kohl, J. Desrosiers, O. B. G. Madsen, M. M. Solomon, and F. Soumis. 2-path cuts for the vehicle routing problem with time windows. *Transportation Science*, 33(1):101–116, 1999. doi: 10.1287/trsc.33.1.101.

[25] J. Lysgaard, A. N. Letchford, and R. W. Eglese. A new branch-and-cut algorithm for the capacitated vehicle problem. *Mathematical Programming*, 100(2):423–445, 2004. doi: 10.1007/s10107-003-0481-8.

[26] M. E. LÃ$\frac{1}{4}$bbecke and J. Desrosiers. Selected topics in column generation. *Operations Research*, 53(6):1007–1023, 2005. doi: 10.1287/opre.1050.0234.

[27] G. L. Nemhauser and L. A. Wolsey. *Integer and Combinatorial Optimization*. John Wiley & Sons, Inc., 1988.

[28] B. Petersen, D. Pisinger, and S. Spoorendonk. Chvátal-Gomory rank-1 cuts used in a Dantzig-Wolfe decomposition of the vehicle routing problem with time windows. In B. Golden, R. Raghavan, and E. Wasil, editors, *The Vehicle Routing Problem: Latest Advances and New Challenges*, pages 397–420. Springer, 2008. doi: 10.1007/978-0-387-77778-8_18.

[29] G. Righini and M. Salani. Symmetry helps: bounded bi-directional dynamic programming for the elementary shortest path problem with resource constraints. *Discrete Optimization*, 3(3):255–273, 2006. doi: 10.1016/j.disopt.2006.05.007.

[30] G. Righini and M. Salani. New dynamic programming algorithms for the resource constrained shortest path problem. *Networks*, 51(3):155–170., 2008. doi: 10.1002/net.20212.

[31] M. Salani. *Branch-and-Price Algorithms for Vehicle Routing Problems*. PhD thesis, Universitá Degli Studi Di Milano, Facoltá di Scienza Matematiche, Fisuche e Naturali Dipartimento di Technologie dell'Informazione, Milano, Italy, 2005.

[32] M. M. Solomon. Algorithms for the vehicle routing and scheduling problem with time window constraints. *Operations Research*, 35(2):234–265, 1987. doi: 10.1287/opre.35.2.254.

[33] P. Toth and D. Vigo. An overview of vehicle routing problems. In P. Toth and D. Vigo, editors, *The Vehicle Routing Problem*, chapter 1, pages 1–26. SIAM, 2002.

[34] F. Vanderbeck. On Dantzig-Wolfe decomposition in integer programming and ways to perform branching in a branch-and-price algorithm. *Operation Research*, 48(1):111–128, 2000. doi: 10.1287/opre.48.1.111.12453.

Chapter 7

Optimal Routing with Failure Independent Path Protection

Thomas Stidsen
DTU Management Engineering, Technical University of Denmark

Bjørn Petersen
DIKU Department of Computer Science, University of Copenhagen

Simon Spoorendonk
DIKU Department of Computer Science, University of Copenhagen

Martin Zachariasen
DIKU Department of Computer Science, University of Copenhagen

Kasper Bonne Rasmussen
Department of Computer Science, ETH Zurich, Switzerland

Abstract

Reliable communication has become crucial in today's information society. Modern communication networks are required to deliver reliable communication to their customers. Unfortunately, protection against network failures significantly hampers efficient utilization of network investments, because the associated routing problems become much harder. In this paper we present a rigorous mathematical analysis of one of the most promising protection methods: Failure independent path protection. We present an LP model which is solved by column generation. The subproblem is proven to be strongly $\mathcal{NP}$-hard, but still solvable for medium sized networks through the use of specialized dynamic programming algorithms. This enables us to evaluate the performance of failure independent path protection for 8 networks with up to 37 nodes and 57 links. The results indicate that only between 3% and 8% extra network capacity is necessary when compared to the capacity required by complete rerouting (which is the absolute lower bound for single link failure protection).

Published in the journal *Networks*, 2009

Keywords: Failure independent path protection, column generation, protection capacity minimization

1 Introduction

Today's information society relies increasingly on advanced communication networks. This has led to massive investments in increased communication network capacity. In order to utilize these investments the network operators perform *t*raffic engineering, i.e., they route communication to maximize the utilization of the capital invested in the communication network.

Most of the backbone networks which today carry long distance communication traffic use path based routing, i.e., a communication connection between two points in the network is established along one or more fixed paths. Despite the huge success of the packet switched Internet, path based routed network technology will continue to be the dominant technique of backbone networks, because traffic engineering can be performed much more efficiently than in packet switched networks. Examples of such path switched network technologies are SDH/SONET or DWDM networks or circuit switched network technologies like PSTN/ISDN. Furthermore, the new Multi Path Label Switching (MPLS) [34] protocol enables packets to be routed on fixed paths.

The standard model of a path switched communication network is a directed graph $G = (V, A)$ consisting of a set of nodes V and a set of arcs A. The nodes correspond to telecommunication switches. The telecommunication switches route the communication signals through cables. We will assume that all cables enable bidirectional communication and therefore we will model one cable using two arcs, one each way between the end nodes. We assume that a static communication connection demand is given which requires one-way communication between an origin node o_k and a terminating node d_k of volume ρ_k for a set of demands $k \in K$. For each demand k we should construct a single *primary* (or working) path from o_k to d_k, and all the required volume of traffic ρ_k should be sent over this primary path (i.e., traffic should be non-bifurcated).

Communication networks are increasingly required to be *reliable*. If we cannot trust our messages to reach the receiver, the use of a communication network is limited. Communication networks are prone to failures and many different types of failures can occur. Switches (nodes) can lose power, experience software and hardware failures, etc. Cables (arcs) can be cut by entrepreneurs or by natural disasters. For simplicity, in this paper we will only consider single cable failures, i.e., simultaneous failure of the two arcs which correspond to a cable. This is a well-known and widely used simplification [15, 26].

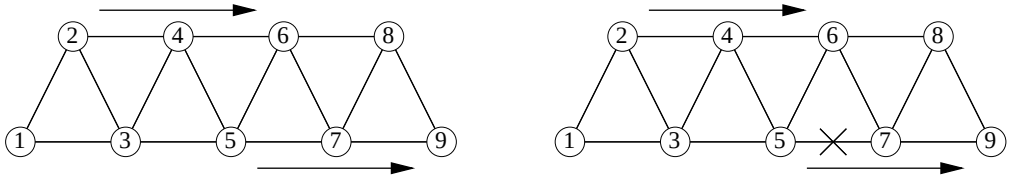
Multiple cable failures can occur in networks, but are less probable. Several cables can fail, if, e.g., a switch fails or a single cable failure in a lower network layer may result in multiple failures in the upper layers. These kind of network errors are of increasing importance but they also make network protection significantly harder, e.g., the problem of finding failure independent paths is $\mathcal{NP}$-complete in the face of multiple cable failures [17].

When a cable fails, the network operator either has to repair the cable or *re-route* the failed paths around the failure. Because repairing a cable can take considerable time, rerouting is an interesting alternative. The main problem with rerouting is that enough capacity needs to be available on the remaining non-failed cables to enable rerouting. Traffic engineering which takes into account the possibility of a cable failure becomes significantly more complex, but

is again important in order to utilize network investments.

In this paper we assume that traffic which is routed along one primary paths is rerouted along the same *backup* (or protection) path. Hence rerouted traffic is non-bifurcated. The cost function is simple: We assume that a linear cost term c_a for using capacity on arc a has to be paid. The required capacity of an arc is the maximum capacity required for all failure situations (the network should be able to accommodate necessary rerouting). The total cost of the network is the sum of costs over all arcs. It should be noted that in our model arcs have no capacity bounds — in contrast to the well-known multi-commodity flow model [1].

In Figure 1(a) two paths are established, from node 2 to node 6 and from node 5 to node 9, both with a volume of 1, that is, $(o_1, d_1, \rho_1) = (2, 6, 1)$ and $(o_2, d_2, \rho_2) = (5, 9, 1)$. In Figure 1(a) — and all the other figures in this paper — we have only drawn the bidirectional cables, and *not* the two corresponding arcs for each cable, in order not to complicate the figures unnecessarily. The necessary capacity of a cable corresponds to the sum of the necessary arc capacities for that cable. Given the paths chosen in Figure 1(a) an arc capacity of 1 is then required on the arcs $(2, 4)$, $(4, 6)$, $(5, 7)$ and $(7, 9)$, resulting in a total required Non-Failure (NF) network capacity of 4. In Figure 1(b) the cable between node 5 and node 7 fails resulting in the failure of arc $(5, 7)$ and arc $(7, 5)$. This results in a communication breakdown for the path from node 5 to node 9.



(a) Two paths (b) A cable break

Figure 1: Path switched routing.

In order to protect communication against a cable failure, a rerouting strategy needs to be planned for each possible cable failure, i.e., a protection method needs to be installed. (Because rerouting methods protect against failures, we will use rerouting methods and protection methods interchangeably.) The importance of network reliability and the importance of minimizing network investments have resulted in a large number of rerouting methods. It is beyond the scope of this paper to review these and we refer the reader to [15] for a recent and comprehensive survey. One of the promising methods is p-cycle protection. This is a clever extension of the well-known ring protection scheme, which significantly improves the capacity requirements necessary for protection [15, 31]. Furthermore, the use of p-cycles enable fast protection of communication, as provided by ring protection. Despite these promising features, p-cycles have not (yet) achieved widespread application.

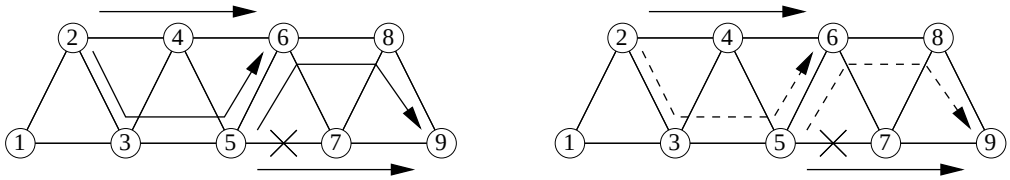
In this paper we will consider traffic engineering optimization methods for the Failure Independent Path Protection (FIPP) method for path switched networks. In this protection method the backup path for a given demand is independent of the failure related to the primary path, i.e., independent of which of the cables in the primary path have failed. This protection method is also called Shared Backup Path Protection in [15] or Global Backup Path Protection in [6].

The outline of the paper is as follows. In Section 2 we give a brief description of different path protection methods. This leads us to focus on the FIPP method for which we give a

mathematical model in Section 3. In the same section we also present a column generation algorithm to solve a relaxed model and discuss the computational complexity of the sub-problem. In Section 4 we then present and discuss the results when applying the column generation algorithm to a number of test cases. In Section 5 we discuss possible extensions and in Section 6 we draw some conclusions.

2 Path protection method

The classic path protection method employed in path switched networks is 1+1 protection. Figure 2(a) shows how the 1+1 protection method can be used to protect the path connections from Figure 1. In 1+1 protection, two cable disjoint paths (and hence arc disjoint paths) are established and actively used. If an arc fails on one path, the other path will survive and enable the receiving node to restore communication by just switching to the other incoming signal. This method is simple, there are well-defined standards, but the required network capacity is always at least twice the required non-failure network capacity. The total network capacity required in the example in Figure 2(a), assuming the same demands, is 10. Notice in particular that a capacity of 2 is required on arc $(5, 6)$.



(a) 1+1 protection (b) FIPP

Figure 2: Capacity sharing illustrated.

2.1 Comparing path protection methods

We now define two measures: Restoration Over Build (ROB) network capacity and Relative Restoration Over Build (RROB) network capacity.

ROB: The extra network capacity necessary to ensure protection, i.e., the network capacity for both routing and protection minus the NF network capacity, assuming shortest path routing. In the example from Figure 2(a), $ROB = 10 - 4 = 6$.

RROB: The relative extra network capacity necessary to ensure protection, i.e., the ROB network capacity divided by the NF network capacity. In the example from Figure 2(a), $RROB = \frac{10-4}{4} = 1.5$, meaning that 1+1 protection in this case costs 150% extra network capacity compared to the necessary non-failure network capacity.

The FIPP method is a slight variation of 1+1 protection: Instead of actively sending data packets on both paths, one path is designated the primary path and only when that path fails will the data packets be sent along the backup path. In Figure 2(b) the same two protected connections as in Figure 2(a) are shown, but now there is a primary path (full line) and a backup path (dashed line) for each path. But the required network capacity has decreased. The arc $(5, 6)$ now only needs a capacity of 1, because the backup paths are not being used

at the same time. This concept is called *sharing* and is possible because we only guarantee protection against single cable failures and because the two primary paths are cable disjoint. For the FIPP method, the NF network capacity is again 4, but the ROB network capacity is now 9, which leads to an RROB network capacity of 1.25.

In order to utilize the path protection methods traffic engineering has to be performed in order to minimize the RROB network capacity. When working with 1+1 protection this is a well-studied problem for which there exist polynomial-time algorithms [4, 33]. This is *not* the case for the FIPP method. Because of the possibility of sharing the capacity for the backup paths, the best choice of primary path and backup path for each end-to-end demand node pair becomes interdependent.

A practical solution to the FIPP traffic engineering problem is studied in [23]. In order to simplify the problem, the dependency between different protected communication connections is ignored in [23]. Instead, the focus is on algorithms which can find pairs of disjoint paths, where the cost of backup paths is assumed to be some constant factor cheaper than the primary paths. Because of the sharing possibility it is reasonable that the capacity costs for each arc of the backup path are less than the capacity costs for each arc of the primary path. Even this simplified problem is $\mathcal{NP}$-hard [23] and a number of different heuristics are suggested to find good, though not optimal, solutions to the problem. This line of research is continued in [22]. It should be emphasized that the cost model for backup paths used in [22, 23] is approximate. We quantify the *exact* relationship between costs for primary and backup paths in Section 3.1 and prove that the resulting optimization problem is strongly $\mathcal{NP}$-hard.

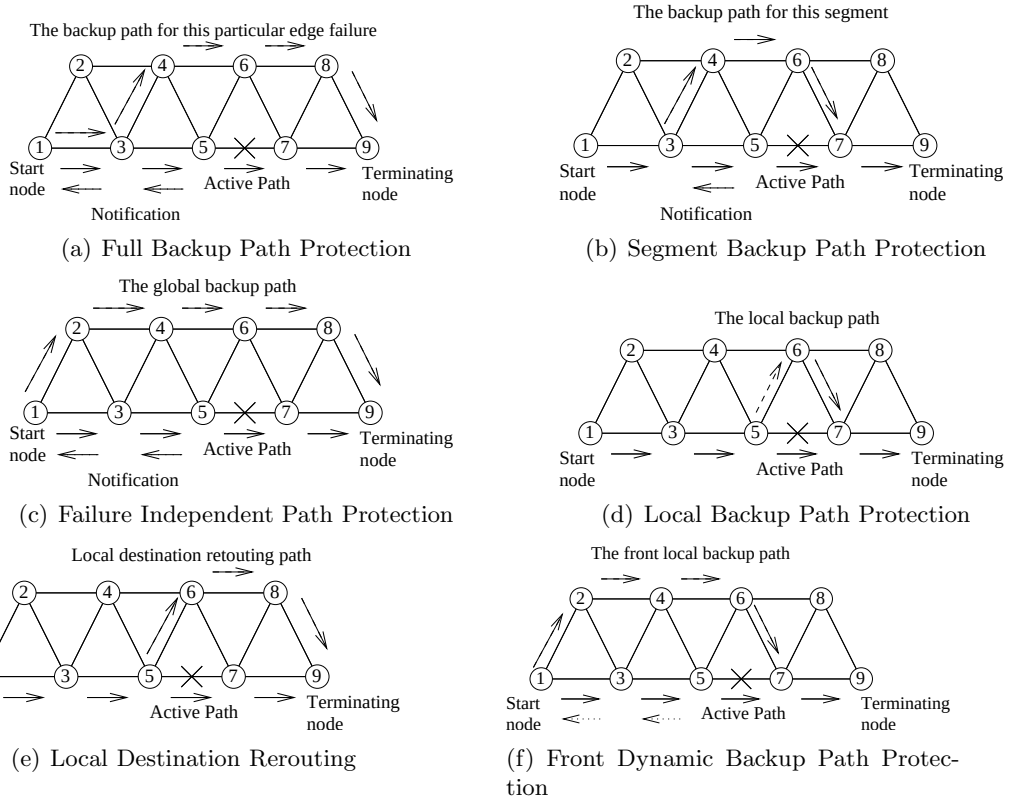
In [26] the full FIPP traffic engineering problem is considered. A column generation approach, similar to the approach in this paper, is considered. The same mathematical model for the column generation master problem is formulated, but the subproblem is *not* formulated. This means that if an optimal solution is required, the full set of disjoint paths has to be pre-generated, and this is only feasible for small networks.

2.2 Different path protection methods

The Failure Independent Path Protection method is just one example of a path protection scheme, and there are a number of other methods. The different path protection methods all use one primary path, but protect the primary path in different ways. In Figure 3, which is (partly) taken from [6], six path protection methods are presented. If the path protection methods are only allowed to choose the backup path based on the failed cable, this list is complete, but a number of additional variations exists, some of which are described in Section 2.3.

Full Backup Path Protection (FBPP)

Theoretically FBPP [24], see Figure 3(a), is the most efficient path protection method. (This method is not included in [6].) Given a primary path, each cable which can fail on the primary path is protected by a unique backup path. There are no limitations regarding these backup paths, except they are, obviously, not allowed to use any of the two failed arcs in the cable which they protect. This gives the highest possible freedom in choosing the cheapest protection paths and all the other path protection methods are more restrictive in the choice of backup paths and hence more costly.



(a) Full Backup Path Protection

(b) Segment Backup Path Protection

(c) Failure Independent Path Protection

(d) Local Backup Path Protection

(e) Local Destination Rerouting

(f) Front Dynamic Backup Path Protection

Figure 3: Different path protection schemes.

Segment Backup Path Protection (SEBPP)

SEBPP, see Figure 3(b), protects segments (sets of cables) of the primary path with the same backup path. Hence several cables in the same segment are forced to share backup paths.

Failure Independent Path Protection (FIPP)

FIPP, see Figure 3(c), limits the choice of backup path even further, such that only one backup path is allowed. This forces the backup path to be cable disjoint with the primary path.

Local Backup Path Protection (LBPP)

LBPP [24], see Figure 3(d), performs a local protection, i.e., the rerouting paths are required to lead from one node of the failed cable to the other node of the failed cable. This resembles the classical span protection, but in this case different reroute paths may be chosen for each connection.

Local Destination Rerouting (LDR)

LDR [2], see Figure 3(e), is a variation of local protection, where the connection paths are rerouted directly to the end node of the connection. LDR preserves the fast rerouting time of Local Backup Path Protection, but is more efficient regarding ROB network capacity.

Front Dynamic Backup Path Protection (FDBPP)

FDBPP, see Figure 3(f), is another variation of local protection, where the connection path is rerouted from the start node to the end node of the failed cable. To the best of our knowledge this type of protection has not been suggested anywhere else and is only included to make the list of path protection methods complete. We do not expect the FDBPP method to be implemented anywhere.

2.3 Further variations

The description of the different path protection schemes is very simplified and a number of variations can be added. Here we briefly mention two of these.

Stub-release is a technique which can be applied to further lower the required network capacity. The idea is that in case of a failed cable, the unharmed parts of the primary path, which are not in use any longer, are released and can be used for protection [25]. Stub-release can improve the capacity efficiency of each method, with the exception of the Local Backup Path Protection method, at the price of a more complicated protection scheme.

To speed up the recovery process, Hashkin protection can be applied [16]. The idea is to loop-back the communication signals at the switch just before the failed cable, to where the backup path starts. Hashkin protection minimize packet-loss, but requires more network capacity and cannot be used in Local Backup Path Protection and Local Dynamic Backup Path Protection.

2.4 Motivation for FIPP

Out of the 6 different types of path protection described in Section 2.2, we only consider the FIPP method in this paper.

FIPP is the only path protection method for which the protection *action* does not depend on which cable actually fails — it is *failure independent*. This makes FIPP the simplest of the path protection methods. Furthermore, the complex switching schemes take place at the start node of the connection path, which may be an advantage in future networks. It is *not* the most capacity efficient path protection method. The most efficient method is FBPP, but FBPP requires administration of a large number of backup paths. Furthermore, in Section 4 we demonstrate that the FIPP method is indeed a *very* efficient protection method, when optimal routing of the primary path and the backup path is performed.

The main disadvantage with the FIPP method is the relatively long restoration time, i.e., the time it takes to restore communication. This is because of the notification time – which is the backward communication time between the node which observes the failure and the node from which the connection paths originates. We have illustrated the notification time by dotted arrows in Figure 3 for the path protection methods for which this is necessary. For a more complete discussion of restoration time, we refer to [6].

3 LP model and column generation approach

In this section we start by defining the FIPP optimization problem formally. Then we present an LP model for a relaxed version of the FIPP optimization problem, the so-called *fractional* FIPP optimization problem. The LP model has an exponential number of variables, and

hence we solve it using column generation. In Section 3.1 we describe the associated pricing problem (or subproblem). A MIP model for solving the subproblem is given in Section 3.2, and in Section 3.3 we show that the subproblem is in fact strongly $\mathcal{NP}$-hard. Finally, in Section 3.4 we give a labeling algorithm for solving the subproblem, and summarize our column generation algorithm in Section 3.5.

Given, as previously defined, a directed graph $G = (V, A)$ with nodes V and arcs A. For each failure situation $s \in S$ we have a set of failed arcs $F_s \subseteq A$. There is a cost c_a for using one unit of capacity of an arc a. We further assume to know a static set of demand node pairs for which protected connections using the FIPP method should be established. A directed connection between an origin node o_k and a terminating node d_k with a volume of ρ_k should be established for each demand $k \in K$. The optimization objective is to minimize the cost of the required capacity when applying the FIPP method to protect the established connections. This means that for each demand a *pair* of directed failure disjoint paths needs to be found: A primary path p^{pri} and a backup path p^{bac}, both connecting node o_k to node d_k. Such a pair of failure disjoint paths is denoted a *path pair* $\pi = (p^{pri}, p^{bac})$. The objective in the FIPP problem is to find a path pair for each demand $k \in K$, such that the total cost of the capacity required is minimized. Note that the capacity required by an arc is the maximum capacity required taken over all failure situations.

Given these definitions we are ready to present an LP model for the fractional FIPP optimization problem. In this problem we allow more than one path pair to accommodate the flow required by a demand. Let P_k be the set of path pairs that can satisfy demand k, that is, the set of primary/backup paths that connect origin node o_k with terminating node d_k. Let $P_k(a) \subseteq P_k$ be the subset of path pairs for which the *primary* path uses arc $a \in A$. Similarly, let $P_k(a, s) \subseteq P_k$ be the set of path pairs for which the *primary* path fails and the *backup* path uses arc $a \in A \setminus F_s$ in failure situation $s \in S$. Finally, let variable λ_π^k denote the amount of communication flow through path pair $\pi \in P_k$, and let variable θ_a denote the capacity required for arc $a \in A$.

FIPP

minimize:

$$\sum_{a \in A} c_a \cdot \theta_a \tag{1}$$

subject to:

$$\sum_{\pi \in P_k} \lambda_\pi^k \geq \rho_k \quad \forall\, k \in K \tag{2}$$

$$\sum_{k \in K} \sum_{\pi \in P_k(a)} \lambda_\pi^k + \sum_{k \in K} \sum_{\pi \in P_k(a,s)} \lambda_\pi^k \leq \theta_a \quad \forall\, s \in S,\ a \in A \setminus F_s \tag{3}$$

$$\lambda_\pi^k, \theta_a \in R_+$$

The objective function is given by (1) and it is the cost of the summed network capacity. The demand constraint (2) ensures that enough capacity is established on the path pairs. The capacity constraint (3) ensures that enough capacity is allocated to route the communication on each arc a in each failure situation s which does not disrupt the arc.

The problem with this LP-model is that the number of path pairs grows exponentially with the network size, and hence the complete model can only be solved for small network sizes. Instead, we will use a column generation algorithm such that only a subset of the path pairs is generated. The optimization subproblem to generate new path pairs with negative reduced costs is given in Section 3.1, and in Section 3.5 the column generation algorithm is given.

It is clear that the fractional FIPP optimization problem is a relaxation of the original FIPP optimization problem which is $\mathcal{NP}$-hard [32]. The hardness of the fractional FIPP optimization problem on the other hand is still an open problem. The LP model can therefore be used for lower bounding in a branch-and-price algorithm for the FIPP optimization problem. The bound can however be weak, because the bound of the *relaxed* FIPP model is equivalent to the bound of the relaxed FBPP model, if the primary paths consists of one link. For primary paths of one link, each of the backup paths for the FBPP model can be constructed by generating path pairs, i.e., the one hop primary path and different backup paths. For primary paths which are not one hop however, the relaxed FIPP model and the relaxed FBPP model are not equivalent, because in the FIPP model the feasible backup paths are more limited than the feasible backup paths for the FBPP model. In other words, it will depend on the network and the communication demand how good a bound the relaxed FIPP model can deliver compared to the bound of the FBPP model.

3.1 Subproblem: Quadratic Cost Disjoint Path Problem

For the master problem for FIPP optimization problem let $\alpha_k \geq 0$, $k \in K$, be the dual variables associated with the (negated version of) constraint (2), and let $\beta_a^s \geq 0$, $s \in S$, $a \in A \setminus F_s$, be the dual variables associated with constraint (3). Our task is to decide if there exists a pair of primary and backup paths $\pi = (p^{pri}, p^{bac})$ from some origin node o_k to some terminating node d_k with negative reduced cost for some $k \in K$.

The reduced cost of a pair of paths (p^{pri}, p^{bac}) is computed as follows. The cost of an arc $a \in p^{pri}$ is $\sum_{s \in S} \beta_a^s$, while the cost of an arc $a \in p^{bac}$ is $\sum_{s \in S: F_s \cap p^{pri} \neq \emptyset} \beta_a^s$. Note the asymmetry in the definition of arc costs in primary and secondary paths: For an arc on the primary path the cost is the sum taken over *all* failure situations, while for an arc on the backup path the sum is only taken over the failure situations that affect an arc on the *primary* path. The total reduced cost of (p^{pri}, p^{bac}) is now

$$-\alpha_k + \overbrace{\sum_{a \in p^{pri}} \sum_{s \in S} \beta_a^s}^{primary\ path\ cost} + \overbrace{\sum_{a \in p^{bac}} \sum_{s \in S: F_s \cap p^{pri} \neq \emptyset} \beta_a^s}^{backup\ path\ cost}$$

The *Quadratic Cost Disjoint Path Problem (QCDPP)* is to compute a pair of paths $\pi = (p^{pri}, p^{bac})$ with minimum total cost. The name of the problem comes from the fact there is a pairwise (or quadratic) dependence on the cost of the backup path as a function of the primary path. Since the dual variables β_a^s are non-negative, there clearly exists an optimal solution where both the primary path p^{pri} and the backup path p^{bac} are simple. Hence in the following we require that the paths p^{pri} and p^{bac} are simple and arc disjoint.

3.2 MIP model for QCDPP

A primary path is defined by the binary variables x_a for all $a \in A$ and a backup path is defined by the binary variables y_a for all $a \in A$. We define the sets $\delta^+(i)$ as the arcs going out of node $i \in V$ and $\delta^-(i)$ as the set of arcs going into node $i \in V$. We again use the set of failed arcs F_s and define the cardinality of the set as $|F_s|$, i.e., the number of arcs which fails in situation $s \in S$. The binary variables u_s for all $s \in S$ detect whether the primary path is interrupted by failure s and the binary variables v_s for all $s \in S$ detect whether the backup path is interrupted by failure s. Furthermore, the auxiliary variables z_s^a for all $s \in S$ and all $a \in A$ detect if the primary path is interrupted by failure s at the same time as the backup path use arc a.

QCDPP

minimize:

$$c^k_{reduced} = -\alpha_k \; + \overbrace{\sum_{a\in A}\sum_{s\in S} \beta_a^s \cdot x_a}^{primary\ path\ cost} \; + \overbrace{\sum_{a\in A}\sum_{s\in S} \beta_a^s \cdot z_s^a}^{backup\ path\ cost} \tag{4}$$

subject to:

$$\sum_{a\in\delta^+(i)} x_a - \sum_{a\in\delta^-(i)} x_a = \begin{cases} 1 & i = o_k \\ -1 & i = d_k \\ 0 & \text{otherwise} \end{cases} \quad \forall\, i \in V \tag{5}$$

$$\sum_{a\in\delta^+(i)} y_a - \sum_{a\in\delta^-(i)} y_a = \begin{cases} 1 & i = o_k \\ -1 & i = d_k \\ 0 & \text{otherwise} \end{cases} \quad \forall\, i \in V \tag{6}$$

$$|F_s| \cdot u_s \geq \sum_{a\in F_s} x_a \quad \forall\, s \in S \tag{7}$$

$$|F_s| \cdot v_s \geq \sum_{a\in F_s} y_a \quad \forall\, s \in S \tag{8}$$

$$u_s + v_s \leq 1 \quad \forall\, s \in S \tag{9}$$

$$z_s^a \geq u_s + y_a - 1 \quad \forall\, s \in S,\ a \in A \tag{10}$$

$$x_a, y_a, u_s, v_s \in \{0,1\}, \qquad z_a^s \in [0,1] \tag{11}$$

The objective function (4) is the reduced cost $c^k_{reduced}$ of the two disjoint paths. The first double sum calculates the costs for the primary path. The second double sum then calculates the cost for the backup paths. Notice that each arc a in the backup path only costs β_a^s in situation s if the primary path is disrupted in failure situation s. This is detected by the variable z_s^a. Finally the dual value α_k from constraint (2) is subtracted to calculate the corresponding reduced cost. Both the primary path variables x and the backup path variables y are constrained to form paths by constraint (5) and (6), respectively. The path disruption variables, u for the primary path and v for the backup path, are set by constraint (7) and (8) respectively. Variables u and v are then used in constraint (9) to ensure failure disjointness of the paths. In constraint (10) the auxiliary variable z_s^a is forced to the value 1 if the primary

path is disrupted in situation s and the backup path uses the arc a. Finally the domains of the variables are given by constraint (11).

We consider two variants of failure situations: In the *single arc* failure variant there is one failure situation for each arc in A. In the *single link* failure variant there is one failure situation for each pair of opposite arcs, i.e., when the corresponding undirected edge is broken.

In Section 3.3 it is proved that the sub-problem above is $\mathcal{NP}$-hard. However, if instead the primary paths were pre-calculated and the task was to find the best usage of the primary paths, at the same time finding the best backup paths, the sub-problem would be a simple shortest path problem (with links of the primary path removed from the network).

3.3 $\mathcal{NP}$-hardness of QCDPP

We now prove that QCDPP is strongly $\mathcal{NP}$-hard for the single arc and single link failure variants. First we present the proof for the single arc variant and then we indicate how this leads to an $\mathcal{NP}$-hardness proof for the single link variant. In the single arc variant the set of failure situations S is identical to the set of arcs A. The decision version of QCDPP with single arc failures is formally defined as follows (where the constant term $-\alpha_k$ in the objective function of QCDPP is ignored).

INSTANCE: Directed graph $G = (V, A)$, pairwise (integer and non-negative) costs β_a^f for all ordered pairs of arcs $(a, f) \in A \times A$, origin node $o_k \in V$, terminating node $d_k \in V$ and integer C.

QUESTION: Does there exist a pair of simple arc disjoint paths $\pi = (p^{pri}, p^{bac})$ from o_k to d_k in G such that

$$\sum_{a \in p^{pri}} \sum_{f \in A} \beta_a^f \; + \; \sum_{a \in p^{bac}} \sum_{f \in p^{pri}} \beta_a^f \; \leq \; C \;\; ?$$

We prove that this problem is $\mathcal{NP}$-complete by reduction from 3-SATISFIABILITY (3SAT) [14]. It is obvious that the decision version of QCDPP is in $\mathcal{NP}$, since given $\pi = (p^{pri}, p^{bac})$ we can compute the corresponding cost and compare it to C in polynomial time.

Let (U, C) be an instance of 3SAT, where $U = (x_1, x_2, \ldots, x_n)$ is a finite set of n variables and $C = (c_1, c_2, \ldots, c_m)$ is a set of clauses where $|c_i| = 3$, $i = 1, \ldots, m$. We assume without loss of generality that each variable appears in at least one clause.

Based on the 3SAT instance we create an instance of the QCDPP with the structure illustrated in Figure 4. The graph consists of two chains of arcs – the so-called top chain and the bottom chain. Two node disjoint paths from o_k to d_k must necessarily have the property that one of the paths travels through the top chain while the other travels through the bottom chain. By assigning costs appropriately, we will force the primary path to use the bottom chain and the backup path to use the top chain.

We will first assume that we seek two *node* disjoint paths from o_k to d_k in this graph. Later we describe how we can modify the graph so that the paths become *arc* disjoint. Furthermore, the graph that is shown is a directed multigraph, and later we also describe how this graph can be transformed into an ordinary directed graph.

The arcs in the top chain are denoted *variable* arcs, while the arcs in the bottom chain are denoted *clause* arcs. For each clause $c_i \in C$ we have 8 parallel arcs, one for each combination of assignments for the three literals; these assignments are denoted 000, 001, 010 etc. As

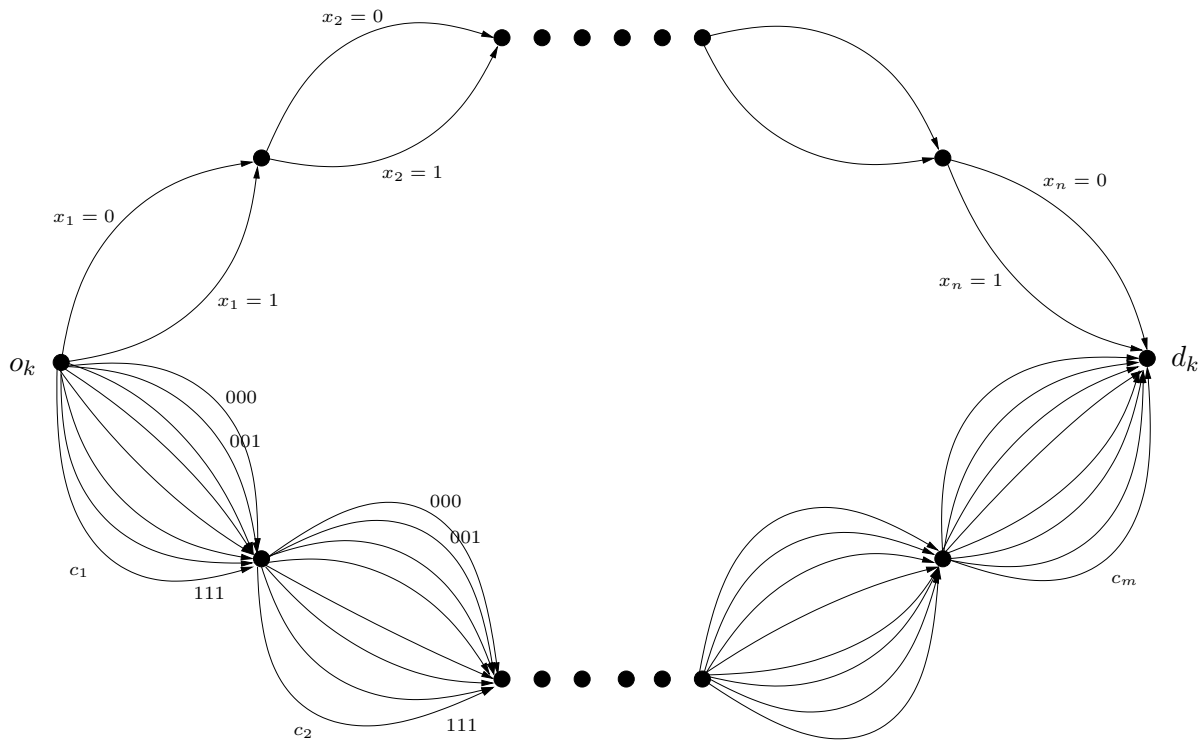


Figure 4: Graph construction for $\mathcal{NP}$-completeness proof.

an example, for the clause $(x_1 \vee x_2 \vee \bar{x}_3)$ the assignment 011 means that $x_1 = 0$, $x_2 = 1$ and $x_3 = 0$. Note that an assignment different from 000 corresponds to a satisfied clause. Similarly, we have two variable arcs for each variable x_j, one arc for $x_j = 0$ and one arc for $x_j = 1$.

We will now assign pairwise costs β_a^f for all ordered pairs of arcs $(a, f) \in A \times A$. We set $\beta_a^f = 0$ for all $(a, f) \in A \times A$ *except from* the following pairs:

- For a *clause* arc a corresponding to the assignment 000 we have $\beta_a^{f'} = 1$ for one arbitrary variable arc f' (say, the arc corresponding to $x_1 = 0$). This means that if the arc a is used by a primary path from o_k to d_k then the cost of a is $\sum_{f \in A} \beta_a^f = 1$.

- For a *variable* arc a and clause arc f, if the variable assignment given by arc a does *not* match the clause assignment given by arc f, then $\beta_a^f = 1$. As an example, the variable arc a corresponding to $x_3 = 1$ has $\beta_a^f = 1$ for the arc f corresponding to the clause $(x_1 \vee x_2 \vee \bar{x}_3)$ with assignment 011. In Table 1 an extended example on how costs are assigned for variable arcs is given.

  Since we assume that each variable appears in at least one clause, each variable edge a has cost at least 1 as a primary edge, since there will be at least one clause assignment that does not match with the variable assignment given by a.

Finally, we set $C = 0$ in the QCDPP instance. Now we prove that we have YES-instance for QCDPP if and only if we have a YES-instance for 3SAT.

Consider a YES-instance for QCDPP, that is, an instance with zero cost. Such an instance must have a primary path p^{pri} following the *clause* arcs from o_k to d_k, since the variable arcs

Assignment	$x_1 = 0$	$x_1 = 1$	$x_2 = 0$	$x_2 = 1$	$x_3 = 0$	$x_3 = 1$
000	0	1	0	1	1	0
001	0	1	0	1	0	1
010	0	1	1	0	1	0
011	0	1	1	0	0	1
100	1	0	0	1	1	0
101	1	0	0	1	0	1
110	1	0	1	0	1	0
111	1	0	1	0	0	1

Table 1: Costs β_a^f associated with variable arcs a for clause f being equal to $(x_1 \vee x_2 \vee \bar{x}_3)$.

have positive costs as primary path arcs. Consequently, the backup path p^{bac} must follow the *variable* arcs from o_k to d_k. Since the total cost of the solution $\pi = (p^{pri}, p^{bac})$ is zero, all arcs of the path p^{pri} correspond to clauses being satisfied (i.e., are different from the clause assignments 000 which have cost 1 as primary path arcs). Also, since the total cost of $\pi = (p^{pri}, p^{bac})$ is zero, the variable arcs followed by p^{bac} match the assignments in the clause arcs. Therefore, assigning the variables x_j, $j = 1, \ldots, n$, to the values indicated by the path p^{bac} gives a satisfying assignment for the 3SAT-instance.

For the other direction, consider a YES-instance for 3SAT. By letting p^{bac} follow the variable arcs in the QCDPP instance as given by a satisfying 3SAT-assignment, and letting p^{pri} follow the clause arcs corresponding to the 3SAT-assignment, we obtain a solution to QCDPP of total cost zero.

By splitting each node in the graph (apart from o_k and d_k) – that is, replacing the node with an arc (u, v), and connecting all in-coming arcs to u and all out-going arcs to v – we force the paths to be *edge* disjoint. Furthermore, the multigraph can be transformed into an ordinary directed graph G by replacing each arc in the multigraph by a sequence of two arcs, and assigning pairwise costs appropriately. Thus we have the following:

Theorem 1 *The decision version of QCDPP when reduced to single* arc *failures is $\mathcal{NP}$-complete even when all pairwise costs are 0 or 1 (and only distinct pairs of arcs can have non-zero costs).*

Consider the directed graph G resulting from the above construction. If, for each arc in G, we add an arc in the opposite direction we obtain a graph G', where bidirectional communication is feasible for each underlying link. Consider the single *link* failure variant of QCDPP for the graph G', where the costs are assigned as in the construction above, but where the β_a^f costs are replaced with β_a^l costs (where l corresponds to a link). Since the primary and backup paths in G' should be simple, no backward arcs in G' will ever be used, and therefore we obtain the following:

Theorem 2 *The decision version of QCDPP when reduced to single* link *failures is $\mathcal{NP}$-complete even when all pairwise costs are 0 or 1 (and non-overlapping pairs of arcs and links can have non-zero costs).*

3.4 Labeling algorithm for the QCDPP

The QCDPP can be formulated as a Shortest-Path Problem with Resource Constraints (SPPRC). The SPPRC is a common subproblem in many graph based problems when using a column generation based algorithm, e.g., the Vehicle Routing Problem with Time Windows [20, 21] and the Crew Pairing Problem [8]. In the following we will shortly define the SPPRC, discuss complexity issues and the application of recent developments within this area, and describe the basic labeling algorithm. Last we will present the reformulation of the QCDPP into an SPPRC.

The SPPRC can be stated as: Given a weighted directed graph $G' = (V', A')$ with nodes V' and arcs A', and a set of resources R. For each node $i \in V'$ and arc $(i, j) \in A'$ there is a weight of each resource $r \in R$ that is determined by a (not necessarily linear but often constant) function, as well as a lower and upper limit on r. For a sub-path in G' there is a resource accumulation of resource $r \in R$ when visiting node i or traversing arc (i, j), i.e., an amount of resource r is accumulated on the path. The total amount of r must respect the lower and upper limits of r in when arriving at node $i \in V'$ or when using arc $(i, j) \in A$. The increase in resource consumption and cost of a path when extended along an arc is defined by a function, that are sometimes denoted *resource extension functions*, see [18]. The objective is to find a minimum cost path from an origin node $o \in V'$ to an destination node $d \in V'$, where the resources satisfy the limits for all resources $r \in R$. In many cases it suffices to have the limits of the resources only at the nodes; in these cases the limits on the edges can be made non-binding.

The SPPRC is $\mathcal{NP}$-hard in the weak sense when the number of resources is a constant and can be solved with dynamic programming based labeling algorithms in pseudo-polynomial time. An extension of the SPPRC is the node elementary version; the elementary shortest path problem with resource constraints (ESPPRC) where paths must be simple. The elementarity constraint can be enforced with the use of a binary resource for each node to indicate if the node is visited on the path and solved as an SPPRC. The ESPPRC is strongly $\mathcal{NP}$-hard, see [11]. However, if G' does not contain negative weight cycles the additional resources can be disregarded since a least weight path that is simple will always exist, hence the problem can be solved in pseudo-polynomial time. Although the reformulation (see details below) of the QCDPP into a SPPRC leads to a graph with no negative weight cycles, the number of resources amounts to one binary resource per failure scenario, i.e., one per two arcs in G for the single link failure case in the QCDPP. That is, the number of resources in the SPPRC depends on the input of the QCDPP, hence the complexity of the labeling algorithm is exponential when regarding the reformulation of the QCDPP. Also, it is important to note that the reformulation of the QCDPP into a SPPRC results in a non-constant extension function where the weight of the arcs on the backup path depend on the failure scenarios that are affected by the primary path.

A comprehensive overview of work related to SPPRC is outside the scope of this article, but we will briefly discuss some recent results. For further details on mathematical models and solution methods we refer the reader to the survey of Irnich and Desaulniers [19]. Dynamic programming based methods denoted *labeling algorithms* are to date the most dominant approach to solving the SPPRC. However, recently Carlyle et al. [7] present a Lagrangian relaxation based method. The approach is applicable for problems with no negative weight cycles and shows good results when few resources are considered. However, due to the nature of the non-constant extension function on the arc weights in our reformulation this approach

is not directly applicable; also we consider a large number of resources which may limit the effect of the Lagrangian relaxation.

Dumitrescu and Boland [12] present an improved preprocessing for the SPPRC (with no negative cost cycles) and embed it into a labeling algorithm. They present resource lower bound calculations using Lagrangian relaxation, hence solving a shortest path problems. Again this approach is not applicable in our case due to the arc weight extension function in our reformulation. Furthermore, this approach have very limited use when only considering binary resources, which is indeed the case for our reformulation, since the resource bounds are already very tight. Feillet et al. [13] address the ESPPRC and propose to consider unreachable nodes instead of visited nodes with the binary resources. The unreachability of a node is determined based on limits on other resources. In our context this would correspond to deciding if a failure scenario cannot be triggered. However, this is difficult to decide without actually visiting the arcs of the scenario, since triggering a scenario does not directly depend on other resources but on the topology of the graph. Therefore the unreachability concept cannot readily be used in our case.

A very successful labeling algorithm by Righini and Salani [27] showed how a significant speedup can be gained by using a bi-directional approach. That is, based on a monotone resource (e.g., the number of nodes on the path) a breaking point is chosen (e.g., when half the nodes have been visited) and the labeling algorithm is run from both sides. By splicing paths starting at the origin node o with a reverse path coming from the destination node d one can construct a full path. For this method to work all extension functions must be reversible which unfortunately is not the case for our objective function. Boland et al. [5] and Righini and Salani [28] independently proposed to relax the state-space of the labeling algorithm such that only a subset of resources are considered to begin with. Any violated resource is then added iteratively until a feasible path has been found. By construction of the graph and the definition of the objective function used in our reformulation, it is doubtful that this approach would perform satisfactory since relaxing resources would yield zero weight arcs in the associated backup path, making it necessary to add resources until all feasible backup paths are covered.

In a labeling algorithm the labels represent partial paths that are extended (using the extension functions) in all feasible directions from the origin node o. Each label L (a vector with $R+1$ components) stores the cost of the partial path $T_{cost}(L)$ and the current value $T_r(L)$ of each resource $r \in R$. To avoid enumerating all feasible paths in G', only Pareto-optimal labels (i.e., labels that are not proved to be dominated by other labels) are kept during the execution of the algorithm. When using non-decreasing extension functions (which is the case for the reformulation of QCDPP), the label dominance criterion can be stated as follows.

Proposition 1 ([9]) *Let L and L' be two labels representing partial paths ending at the same node. Label L dominates label L' (which can be discarded) if*

$$\begin{aligned} &T_{cost}(L) \leq T_{cost}(L') \\ &T_r(L) \leq T_r(L') \qquad\qquad \forall r \in R. \end{aligned}$$

When equality holds for all label components, one of the two labels must be kept. Figure 5 summarizes the concept of a labeling algorithm. The initial state is represented by the label L_o at the starting node. This label is enqueued on a priority queue Q that keeps track of all unprocessed labels. The algorithm runs until all labels have been processed. In each iteration the next label L from Q is dequeued. The set of nodes (FEASIBLE EXTENSION(L)) that

```
Initialize label L_o
ENQUEUE(Q, L_o)
while Q is not empty
   L := DEQUEUE(Q)
   for each node i ∈ FEASIBLE EXTENSION(L)
      L_i := EXTEND LABEL(L, i)
      if i = d
         then ENQUEUE(S, L_i)
         else ENQUEUE(Q, L_i)
   REMOVE DOMINATED(Q)
return S
```

Figure 5: Pseudo-code for labeling algorithm.

are feasible extensions of the partial path represented by L, with regard to connectivity and resource limits, is determined. L is extended to these nodes using the resource extension functions (implemented in EXTEND LABEL(L, i)) to create the new label L_i for node i. If the extended label L_i is extended to the end node d it is stored as a solution in the queue S otherwise L_i is enqueued on Q for future processing. Last Q is cleaned for dominated labels so only Pareto-optimal labels remain.

Next, we consider the transformation of the QCDPP stated as (4)-(11) into a SPPRC Recall the graph $G = (V, A)$ for the QCDPP where a minimum cost primary and backup path pair must be found from $o_k \in V$ to $d_k \in V$ over all $k \in K$. Let $V' = \{i' : i \in V\}$ be a copy of all nodes in V and let $A' = \{(j', i') : (i, j) \in A, i', j' \in V'\}$ be a reversed version of all arcs in A connecting the nodes in V', and let $A''_k = \{(d_k, d'_k) : d_k \in A, d'_k \in A'\}$ be the arc connecting the two node and arc sets for demand pair k. The transformed graph for the kth demand pair is then $G'_k = (V \cup V', A \cup A' \cup A''_k)$ where a primary path will be sought in the first part of the graph with nodes V, then by the arc (d_k, d'_k) the search is switched to the other part of the graph consisting of the nodes V' where a reverse backup path is found. G'_k is illustrated on Figure 6. For each failure situation $s \in S$ it must be ensured that no arcs from F_s is used on the backup path if any of the arcs in F_s was used on the primary path. A binary resource is added for each failure situation $s \in S$. Hence, the set of resources have size $|S|$. Let a label L consist of $1 + |S|$ components, $T_{cost}(L)$ to store the cost of the path and $T_s(L)$ for $s \in S$ to store the bit value of the failure situation resources. $T_s(L)$ will be set to one if the failure scenario s is triggered on the primary path, and resource limits are enforced on the arcs when extending labels. The upper bound for resource $s \in S$ when extending a label on arc a' are given as 0 for $a' \in A' \wedge a \in F_s$ and 1 otherwise. That is, a label L cannot be extended on arc $(i', j') \in A'$ with $(j, i) \in F_s$ for $s \in S$ on the backup path if arc $a \in F_s$ is used on the primary path, i.e., the resource value $T_s(L) = 1$ and the upper bound for s on (i', j') is 0. Hence, in Figure 5 the end node of a is not in the set FEASIBLE EXTENSION(L). Recall that the cost of the backup path depends on the arcs used on the primary path and that $\beta^s_a \geq 0$ and $\alpha_k \leq 0$. The extension along an arc a of a label L (implemented in EXTEND

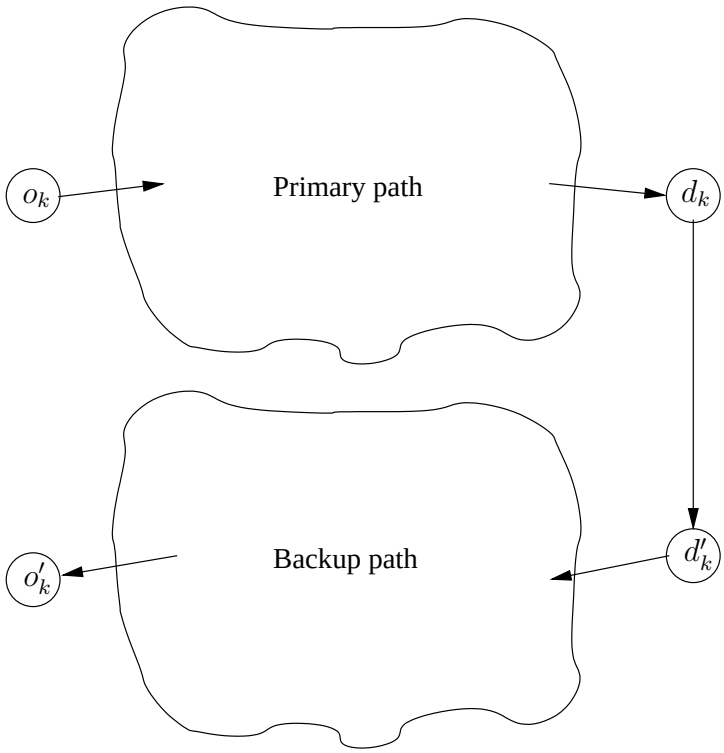


Figure 6: The transformed graph for the kth demand pair. The backup path part of the graph is a reversion of the primary path part, i.e., the path found is a forward directed primary path and a reversed backup path.

LABEL(L, i)) proceeds as follows to create a new label L':

$$T_{cost}(L') = T_{cost}(L) + \begin{cases} \sum_{s \in S} \beta_a^s & a \in A \\ \sum_{s \in S: T_s(L)=1} \beta_{(j,i)}^s & a = (i', j') \in A' \\ -\alpha_k & a \in A''_k \end{cases}$$

$$T_s(L') = \begin{cases} 1 & a \in F_s \\ \max\{T_s(L), 0\} & \text{otherwise} \end{cases} \qquad s \in S$$

Both extension functions are non-decreasing, hence the dominance criterion of Proposition 1 can be applied in the labeling algorithm. For the kth pricing problem; a path represented by label L ending in o'_k have the cost:

$$c^k_{reduced} = -\alpha_k \; + \overbrace{\sum_{a \in A(L)} \sum_{s \in S} \beta_a^s}^{primary\ path\ cost} \; + \overbrace{\sum_{a=(i',j') \in A'(L)} \sum_{s \in S: T_s(L)=1} \beta^s_{(j,i)}}^{backup\ path\ cost} \tag{12}$$

where $A(L)$ and $A'(L)$ are the set of arcs used in A and A' respectively. Minimizing expression (12) is equivalent to the objective function stated in (4) and the path found by the labeling algorithm can trivially be split into a primary and a backup path.

```
Initialize α_k , β_a^s
k = 1
do
   k' := k
   do
      SOLVE QCDPP(k, α_k , β_a^s)
      k := k + 1
   while c_reduced^{p,k} ≥ 0 and k' ≠ k
   Update set of path pairs
   SOLVE FIPP with new set of path pairs
   Update α_k , β_a^s
while k' ≠ k
```

Figure 7: Column Generation algorithm.

3.5 Column Generation Algorithm

Given the LP model in Section 3 we can now apply column generation to solve the model, where the subproblem described in Section 3.2 is either solved using a MIP solver or the labeling algorithm described in Section 3.4. Below we briefly describe the column generation algorithm (Figure 7).

In the column generation algorithm in Figure 7 we first initialize α_k and β_a^s with artificial values: $\alpha_k = \sum_{a \in A} c_a$ and $\beta_a^s = \frac{c_a}{|S|}$ (where S is the set of failure situations). This means that it is always profitable to include a path pair of primary and backup paths for each demand k. After entering the main loop, promising path pairs are found based on the current values of α_k and β_a^s. The resulting paths are then added to the set of path pairs and the master problem is solved with the new set of path pairs. This process continues until no negative reduced-cost path pair for any demand can be found.

4 Results

In this section the efficiency of the FIPP protection method is tested on 8 different networks. Basic network data for the 8 networks is given in Table 2. We have chosen to use the simple demand matrix $D^{kl} = 1$ for each pair of nodes.

In Table 3 and Table 4 we compare the computation times when the QCDPP subproblem is solved using the SPPRC labeling algorithm and a standard MIP solver, respectively.

It can be seen from Table 3 and Table 4 that the SPPRC labeling algorithm is significantly faster on all tested networks. Furthermore, two of the networks, Norway and Ta1, cannot be solved using the MIP solver due to excessive memory consumption.

Given the column generation algorithm, we are now able to calculate the optimal protection capacity required for *relaxed* FIPP protection (Table 5). We find the results in Table 5 interesting because it shows how efficient the relaxed FIPP method is. The FIPP method use at most 8% extra network capacity compared to the theoretical lower bound achieved using Complete Rerouting [30] and on average only 4% extra network capacity. We acknowledge that this is only part of the story and that the moment the demands are required to be integer,

Network	Nodes	Edges	Avg. Node Degree	No. Demands
Cost239 [3]	11	26	4.73	55
Europe	13	21	3.23	78
Newyork [29]	16	49	6.12	120
Ta1 [29]	24	51	4.25	276
FranceSND [29]	25	45	3.60	300
Norway [29]	27	51	3.77	351
USA [10]	28	45	3.21	378
Cost266 [29]	37	57	3.08	666

Table 2: Tested networks and their characteristics.

Network	Rows	Columns				Iter	Time		
		Initial	Final	PerIt	PerDem		Total	CG	CGPct
Cost239 [3]	705	81	1451	42.81	0.78	32	11	1	4.56
Europe [29]	498	99	470	46.38	0.59	8	1	1	36.36
Newyork [29]	2472	169	5292	47.44	0.40	108	2438	1875	76.94
Ta1 [29]	2826	327	4013	43.88	0.16	84	17612	17385	98.71
FranceSND [29]	2280	345	2944	57.76	0.19	45	235	191	81.29
Norway [29]	2901	402	3704	58.96	0.17	56	1177	967	82.22
USA [10]	2358	423	3076	60.30	0.16	44	156	77	49.65
Cost266 [29]	3858	723	6516	62.29	0.09	93	2050	1051	51.29

Table 3: SPPRC labeling algorithm results. Rows: Number of rows in LP. Initial: Initial number of master problem columns. Final: Final number of master problem columns. PerIt: Number of columns added per iteration. PerDem: Number of columns added per iteration per demand. Iter: Number of column generation iterations. Total: Total running time in seconds. CG: Total column generation running time in seconds. CGPct: Column generation (label) solve time as percentage of total time.

i.e., that for each demand the entire communication flow is routed on the same primary path and the same backup path, the ROBB is going to increase.

5 Future Research

The mathematical model we on which we base our results is by choice constructed to be as simple as possible. A number of additional model features can be incorporated into the model and some of these may certainly change the above conclusions. In this section we will briefly describe the two model refinements which we regard as the most important.

Firstly, in the current model we consider the demands as a volume of communication ρ_k to be established between two nodes in the network. In the fractional FIPP problem this volume may be divided between a number of path pairs and this is probably not desirable for the communication customers. Instead, each *customer* should be offered one path pair with a certain volume of traffic — corresponding to the original FIPP problem. For the model

Network	Rows	Columns				Iter	Time		
		Initial	Final	PerIt	PerDem		Total	CG	CGPct
Cost239 [3]	705	81	677	1.00	0.02	597	154	145	93.77
Europe [29]	498	99	307	1.00	0.01	209	31	31	98.36
Newyork [29]	2472	169	2328	1.00	0.01	2160	6491	5943	91.56
Ta1 [29]	2826	-	-	-	-	-	-	-	-
FranceSND [29]	2280	345	1408	1.00	0.00	1064	9434	9356	99.17
Norway [29]	2901	-	-	-	-	-	-	-	-
USA [10]	2358	423	1532	1.00	0.00	1110	2406	2304	95.77
Cost266 [29]	3858	-	-	-	-	-	-	-	-

Table 4: MIP results. Rows: Number of rows in LP. Initial: Initial number of master problem columns. Final: Final number of master problem columns. PerIt: Number of columns added per iteration. PerDem: Number of columns added per iteration per demand. Iter: Number of column generation iterations. Total: Total running time in seconds. CG: Total column generation running time in seconds. CGPct: Column generation (MIP) solve time as percentage of total time.

Network	NP capacity	CR RROB	FIPP RROB	Difference
Cost239	86	0.13	0.19	0.06
Europe	158	0.57	0.65	0.08
Newyork	412	0.19	0.24	0.05
Ta1	733	0.76	0.78	0.02
FranceSND	9825	0.66	0.67	0.01
Norway	61	0.59	0.61	0.02
USA	1273	0.50	0.55	0.05
Cost266	14587	0.62	0.64	0.02
Avg.		0.50	0.54	0.04

Table 5: FIPP protection method comparison. NP capacity: Non-Protected required network capacity. CR RROB: Complete Rerouting [30] required network capacity relative to NP capacity. FIPP RROB: FIPP required network capacity relative to NP capacity. Difference: Absolute difference between RROB for CR and FIPP.

presented in Section 3, this results in more variables, and furthermore, these variables have to be *binary* variables. Hence, to solve this model to optimality, a branch-and-price optimization algorithm is necessary.

Secondly, in the current model there is no bound on the capacity θ_a of an arc $a \in A$. In real-life applications, capacities are acquired in *modular* amounts and economies of scale can be modeled. Modular capacities can be included into the model by changing the right hand side of constraint (3) to a sum of integer variables, as shown in the modified constraint (13) below:

$$\sum_{k \in K} \sum_{\pi \in P_k(a)} \lambda_\pi^k + \sum_{k \in K} \sum_{\pi \in P_k(a,s)} \lambda_\pi^k \;\leq\; \sum_m C_m \cdot \theta_{a,m} \quad \forall\, s \in S,\; a \in A \setminus F_s$$

Here the capacity variables $\theta_{a,m} \in Z^+$ correspond to different types of connections, each

possessing a capacity C_m. The objective function is then modified to include different prices for each type of technology. The price pr. capacity unit reflect the economies of scale.

6 Conclusion

In this paper we presented an LP model for the fractional Failure Independent Path Protection (FIPP) optimization problem. The LP model was solved using column generation. We analyzed the subproblem, proved it to be strongly $\mathcal{NP}$-hard and devised a labeling algorithm for solving the subproblem more efficiently. Finally, we evaluated the capacity efficiency of the FIPP method on a number of network instances. The results indicate that the FIPP method appears to be a very efficient protection method — on average only requiring 4% more network capacity than complete rerouting, the absolute lower bound for single link failure protection.

References

[1] R. K. Ahuja, J. B. Orlin, and T. L. Magnanti. *Network Flows: Theory, Algorithms, and Applications*. Prentice Hall, 1993.

[2] J. Anderson, B.T. Doshi, S. Dravida, and P. Harshavardhana. Fast restoration of atm networks. *IEEE Journal on Selected Areas in Communications*, 12(1):128–138, 1994. ISSN 07338716.

[3] P. Batchelor, B. Daino, P. Heinzmann, D. R. Hjelme, P. Leuthold, R. Inkret, G. De Marchis, H. A. Jager, F. Matera, M. Joindot, B. Mikac, A. Kuchar, H.-P. Nolting, E. Coquil, J. Spath, F. Tillerot, B. Caenegem, N. Wauters, and C. Weinert. Study on the implementation of optical transparent transport networks in the european environment-results of the research project COST 239. *Photonic Network Communication*, 2(1):15–32, 2000. ISSN 1387974X. doi: 10.1023/A:1010050906938.

[4] R Bhandari. *Survivable Networks - Algorithms for Diverse Routing*. Kluwer, 1999.

[5] N. Boland, J. Dethridge, and I. Dumitrescu. Accelerated label setting algorithms for the elementary resource constrained shortest path problem. *Operation Research Letters*, 34 (1):58–68, 2006. doi: 10.1016/j.orl.2004.11.011.

[6] E. Calle, J. L. Marzo, and A. Urra. Protection performance components in MPLS networks. *Computer Communications*, 27(12):1220–1228, 2004. ISSN 01403664. doi: 10.1016/j.comcom.2004.02.025.

[7] W.M. Carlyle, J.O. Royset, and R.K. Wood. Lagrangian relaxation and enumeration for solving constrained shortest-path problems. *Networks*, 51(3):155–170, 2008. doi: 10.1002/net.20212.

[8] G. Desaulniers, J. Desroisers, Y. Dumas, S. Marc, B. Rioux, M. M. Solomon, and F. Soumis. Crew pairing at Air France. *European Journal of Operations Research*, 97:245 – 259, 1997. doi: 10.1016/S0377-2217(96)00195-6.

[9] G. Desaulniers, J. Desrosiers, J. Ioachim, I. M. Solomon, F. Soumis, and D. Villeneuve. A unified framework for deterministic time constrained vehicle routing and crew scheduling problems. In T. G. Crainic and G. Laporte, editors, *Fleet Management and Logistics*, pages 57–93. Kluwer, 1998.

[10] J. Doucette, D. He, W. D. Grover, and O. Yang. Algorithmic approaches for efficient enumeration of candidate p-cycles and capacitated p-cycle network design. In *Proceedings. Fourth International Workshop on Design of Reliable Communication Networks, 2003. (DRCN 2003)*, pages 212–220. IEEE, 2003. ISBN 0780381181. doi: 10.1109/DRCN.2003.1275359.

[11] M. Dror. Note on the complexity of the shortest path models for column generation in VRPTW. *Operations Research*, 42(5):977–979, 1994. doi: 10.1287/opre.42.5.977.

[12] I. Dumitrescu and N. Boland. Improved preprocessing, labeling and scaling algorithms for the weight-constrained shortest path problem. *Networks*, 42(3):135–153, 2003. doi: 10.1002/net.10090.

[13] Dominique Feillet, Pierre Dejaxa, Michel Gendreaua, and Cyrille Gueguen. An exact algorithm for the elementary shortest path problem with resource constraints: Application to some vehicle routing problems. *Networks*, 44(3):216–229, 2004. ISSN 0028-3045. doi: 10.1002/net.v44:3.

[14] M. R. Garey and D. S. Johnson. *Computers and Intractability. A guide to the theory of NP-completeness*. Freeman, 1979.

[15] W. D. Grover. *Mesh-Based Survivable Networks*. Prentice Hall PTR, 2004.

[16] D. Haskin and R Krishnan. A method for setting and alternative label switch path to handle fast reroute. Technical report, Internet Engineering Task Force, 2002.

[17] Jian Qiang Hu. Diverse routing in optical mesh networks. *IEEE Transactions on Communications*, 51(3):489–494, 2003. ISSN 00906778. doi: 10.1109/TCOMM.2003.809779.

[18] S. Irnich. Resource extension functions: Properties, inversion, and generalization to segments. *OR Spectrum*, 30(1):113–148, 2008. doi: 10.1007/s00291-007-0083-6.

[19] S. Irnich and G. Desaulniers. Shortest path problems with resource constraints. In G. Desaulniers, Jacques Desrosiers, and M.M. Solomon, editors, *Column Generation*, chapter 2, pages 33–65. Springer, 2005. doi: 10.1007/0-387-25486-2_2.

[20] S. Irnich and D. Villeneuve. The shortest path problem with resource constraints and k-cycle elimination for $k \geq 3$. *INFORMS Journal on Computing*, 18:391–406, 2006. doi: 10.1287/ijoc.1040.0117.

[21] M. Jepsen, B. Petersen, S. Spoorendonk, and D. Pisinger. Subset-row inequalities applied to the vehicle-routing problem with time windows. *Operations Research*, 56(2):497–511, 2008. doi: 10.1287/opre.1070.0449.

[22] P. Laborczi and T. Cinkler. Efficient algorithms for physically-disjoint routing in survivable GMPLS/ASTN networks. In *11th International Telecommunications Network*

Strategy and Planning Symposium. (NETWORKS 2004), pages 185–192. IEEE, 2004. ISBN 3800728400. doi: 10.1109/NETWKS.2004.1341839.

[23] P. Laborczi, J. Tapolcai, P.-H. Ho, T. Cinkler, A. Recski, and H. T. Mouftah. Algorithms for asymmetrically weighted pair of disjoint paths in survivable networks. In T. Cinkler, editor, *Third International Workshop on Design of Reliable Communication Networks, 2001. (DRCN 2001)*, pages 220–227. BME - Budapest Univ. Technol. & Econ, 2001.

[24] Jean François Maurras and Sonia Vanier. Network synthesis under survivability constraints. *4OR*, 2(1):53–67, 2004. doi: 10.1007/s10288-003-0025-3.

[25] S. Orlowski. Local and global restoration of node and link failures in telecommunication networks. Master's thesis, Technische Universität Berlin, February 2003. URL `http://www.zib.de/orlowski/Orlowski2003.pdf.gz`.

[26] M. Pioro and D. Medhi. *Routing, Flow, and Capacity Design in Communication and Computer Networks*. Elsevier, 2004.

[27] G. Righini and M. Salani. Symmetry helps: bounded bi-directional dynamic programming for the elementary shortest path problem with resource constraints. *Discrete Optimization*, 3(3):255–273, 2006. doi: 10.1016/j.disopt.2006.05.007.

[28] Giovanni Righini and Matteo Salani. New dynamic programming algorithms for the resource constrained elementary shortest path problem. *Networks*, 51(3):155–170, 2008. ISSN 0028-3045. doi: 10.1002/net.v51:3.

[29] SNDlib. SNDlib 1.0–Survivable network design data library, 2005. URL `http://sndlib.zib.de`.

[30] T. Stidsen and P. Kjærulff. Complete rerouting protection. *Journal of Optical Networking*, 5(6):481–492, 2006. doi: 10.1364/JON.5.000481.

[31] T. Stidsen and T. Thomadsen. Joint routing and protection using p-cycles. Technical report, Technical University of Denmark, 2005. URL `http://www2.imm.dtu.dk/pubdb/p.php?3939`.

[32] T. Stidsen, M. Kiese, B. Petersen, S. Spoorendonk, and M. Zachariasen. Network capacity planning with shared protection. Work in progress, 2008.

[33] J.W. Suurballe. Disjoint paths in a network. *Networks*, 4(2):125–145, 1974. doi: 10.1002/net.3230040204.

[34] G. Swallow and L. Andersson. MPLS working group, 2003. URL `http://www.ietf.org/html.charters/mpls-charter.html`.

Part III

Conclusion

Chapter 8

Conclusion

Bjørn Petersen
DTU Management Engineering, Technical University of Denmark

1 Summing Up

The main focus of this thesis is on shortest path problems and how to solve them in the context of column and cut generation algorithms. It has been investigated how to solve various forms of resource constrained shortest path problems. The emphasis has been on difficult versions of this problem, namely with the presence of negative weight cycles and costs not directly mappable to the edge weights. These problems appear in a column generation context when handling effects of cutting planes derived from the master problem formulation.

It has been shown theoretically and experimentally how to apply the general purpose mixed integer programming (MIP) cutting planes known as Chvátal-Gomory cuts of rank 1 to the master problem formulation of a Dantzig-Wolfe decomposition of the Vehicle Routing Problem with Time Windows (VRPTW). Furthermore, it has been shown how to incorporate this into a dynamic programming algorithm for the subproblem. Investigations of how cutting planes impact the subproblems complexity, the quality of the lower bounds for the master problem, and the overall running time of Branch-Cut-and-Price (BCP) algorithms have been performed.

It has been shown how to solve the Elementary Shortest Path Problem with Capacity Constrains (ESPPCC) by the use of a Branch-and-Cut algorithm. It has also been shown how alternative reformulations of the Elementary Shortest Path Problem with Resource Constrains (ESPPRC), the Capacitated Vehicle Routing Problem (CVRP), and the VRPTW can be obtained through the use of Partial Paths, so that the difficult part of problems is targeted and movement of complexity between master and pricing problems is facilitated. Finally, an example of how to utilize resource constrained shortest paths in a telecommunication context has been presented.

Experimental results are reported for the VRPTW, the CVRP, the ESPPCC, the ESPPRC, and the problem of finding Optimal Routing with Failure Independent Path Protection.

In Chapter 3 and Chapter 4 it has been shown how the Chvátal-Gomory cuts of rank 1 can be applied to a decomposition model of the VRPTW. In the former chapter, it was shown how a small subset of the Chvátal-Gomory cuts of rank 1, denoted subset-row inequalities, can

be applied to the Set Partitioning master problem, and how to incorporate their dual costs into the pricing problem (the ESPPRC) with the use of additional resources. At the time of publication this algorithm was the most successful exact solution method for the VRPTW. In the latter chapter, these results were extended to include all Chvátal-Gomory cuts of rank 1. However, a slightly more complicated dominance criterion made the pricing problem a bit harder to solve. Running times could not be improved compared to the former approach, but on all problem instances successfully considered it was possible to close the integrality gap completely in the root node.

In Chapter 5 a Branch-and-Cut algorithm for solving the (ESPPCC) was introduced. A compact mathematical model and valid inequalities developed for the ESPPCC were presented as were experimental results on benchmark instances from the literature and on a new set of hard instances. Chapter 6 presented a new decomposition algorithm for Vehicle Routing Problems based on the concept of partial paths where the routes are found by combining smaller sub-routes. Chapter 7 showed in a real world example, namely Optimal Routing with Failure Independent Path Protection, how resource constrained shortest paths problems are useful in a completely different context.

2 Concluding Remark

Considering the successful work with the subset-row inequalities on the VRPTW and the less successful work with the Chvátal-Gomory rank 1 cuts on the VRPTW, it can be concluded that you need to be careful when choosing which cutting planes to include for a given problem. Do not get disencouraged by making subproblems harder, but do not overdo it. It appears that the pricing problem of the decomposed problem should be hard to solve before applying the cutting planes for the master problem. Most likely, the best results would be achieved if the pricing problem is strongly $\mathcal{NP}$-hard to begin with. Also, for the master problem based cutting planes to be effective it is preferable to have a large integrality gap. Otherwise a few quick branches could just as easy close it.

For some kinds of ESPPRC, e.g., the ESPPCC, it appears that labeling algorithms are clearly outperformed by Branch-and-Cut based algorithms. It must be remarked, though, that labeling algorithms sometimes are used in a context where finding several "good" solutions are desired. Labeling algorithms are superb at this but it is not a property that the Branch-and-Cut based algorithms excel at.

Reformulating with partial paths make it possible to balance the running time of the pricing problem against the tightness of the lower bound. It has been shown in theory that both weaker and stronger root bounds can be obtained compared to models with full paths.

Labeling based algorithms can be parallelized with one thread per node of the graph on which the paths are defined. Due to the overhead of handling multiple threads the parallelized code works best when instances are hard. A general framework can be used for different problems solely by changing the functions that define extensions, dominance, and search structures.

3 Future Research

There are many different ways of dividing into partial paths. Striving for a strong bound in the master problem and for an easy pricing problem are conflicting goals. To find the right

division whereby obtaining a good compromise between both goals demands testing many alternatives or possessing lots of luck. Implementation of a column generation algorithm requires some coding and testing both of which are time consuming.

The transportation problems faced by many companies can in general terms be stated as transporting an amount of commodities between a number of locations by some means of transportation. There are typically restrictions associated with the use of different vehicles, e.g., capacity or availability. Furthermore, there may be restrictions on handling the commodities such as specific times for sending and receiving shipments. When optimizing the solution process of a transportation problem, typical objectives are to minimize overall travel cost or time. These problems are basically contained in CVRP and VRPTW. An often overlooked factor in current solution methods is the important concept of uncertainty, both during transportation and in demand and availability of commodities. These stochastic elements are much less studied than their deterministic counterparts. A future research goal could be to investigate how to handle these stochastic events.

Chapter 9

Summery in Danish

Bjørn Petersen
DTU Management Engineering, Technical University of Denmark

1 Resumé

Det primære fokus for denne ph.d.-afhandling har været på korteste vej problemer og hvorledes de løses i forbindelse med kolonnegenereringsalgoritmer. Det er blevet undersøgt hvordan diverse former for resourcebegrænsede korteste vej problemer kan blive løst. Vægten er blevet lagt på svære udgaver af problemet; mere specifikt når kredse med negativ vægt og omkostninger, der ikke kan afspejles direkte på kanterne, har været tilstedeværende. Disse problemer viser sig i kolonnegenereringssammenhænge, når de duale omkostninger fra snitplan i masterproblemet skal behandles.

Det er blevet vist teoretisk såvel som eksperimentelt, hvorledes generelle mixed integer programming (MIP) snitplan af typen Chvátal-Gomory rank 1 kan anvendes på masterproblem formuleringen af en Dantzig-Wolfe dekomponering af ruteplanlægningsproblemet med tidsvinduer (VRPTW). Endvidere er det blevet vist, på hvilken vis dette kan indarbejdes i en dynamisk programmerings algoritme til løsning af delproblemerne. Undersøgelser af hvordan snitplan influerer delproblemernes kompleksitet, kvaliteten af de nedre grænser i master-problemet og den overordnede køretid for Branch-Cut-and-Price (BCP) algoritmer er blevet udført.

Det er blevet vist, hvordan det simple kortestevejproblem med kapacitetsbegrænsninger (ESPPCC) kan løses vha. en Branch-and-Cut algoritme. Det er ligeledes blevet vist hvordan forskellige reformuleringer af det simple kortestevejproblem med ressourcebegrænsninger (ESPPRC), det kapacitetsbegrænsede ruteplanlægningsproblem (CVRP) og VRPTW kan opnås ved at benytte delveje, således at den svære del af problemer er berørt og flytning af kompleksitet mellem master- og delproblem er muliggjort. Til slut er et eksempel på hvorledes resourcebegrænsede kortestevejproblemer kan blive benyttet i forbindelse med telekommunikation blevet præsenteret.

Eksperimentelle resusultater er blevet rapporteret for VRPTW, CVRP, ESPPCC, ESPPRCog problemet med at finde en optimal rutning med Failure Independent Path Protection.

I kapital 3 og kapitel 4 er det blevet vist, hvordan snitplan af typen Chvátal-Gomory rank 1 kan blive anvendt på en delkomponeringsmodel af VRPTW. I det første af disse kapitler

blev det vist, hvordan en lille delmængde af disse snitplan kaldet subset-row uligheder kan blive benyttet på set-partitioning master-problemet, og hvordan deres duale omkostninger bliver håndteret vha. ekstra ressourcer i delproblemet – et ESPPRC. Denne algoritme var da den blev publiseret den mest succesfulde eksakte løsningsmetode for VRPTW. I det andet af kapitlerne er disse resultater blevet udvidet til at inkludere alle snitplan af typen Chvátal-Gomory rank 1. Et lidt mere kompliceret dominanskriterie gjorde dog delproblemet en smule vanskeligere at løse. Køretider kunne ikke forbedres sammenlignet med den første fremgangsmåde, men for alle probleminstanser, der blev betragtet med succes, var det muligt at lukke heltalsgabet fuldstændigt i rodknuden.

I kapitel 5 blev en Branch-and-Cut algoritme til løsning af ESPPCC introduceret. En kompakt model af og lovlige uligheder til ESPPCC blev præsenteret, ligesom eksperimentelle resultater på testinstanser fra litteraturen og et nyt sæt svære instanser blev det. Kapitel 6 præsenterede en ny dekomponeringsalgoritme til ruteplanlægningsproblemer baseret på delvejs-konceptet, hvor ruterne er fundet ved at kombinere mindre delruter. Kapitel 7 viste med et eksempel fra den virkelige verden (optimal rutning med Failure Independent Path Protection), hvordan resourcebegrænsede kortestevejproblemer er brugbare i anderledes sammenhænge.

Part IV

Other Contributions

Chapter 10

The Simultaneous Vehicle Scheduling and Passenger Service Problem Conditionally

Hanne L. Petersen
DTU Transport, Technical University of Denmark

Allan Larsen
DTU Transport, Technical University of Denmark

Oli B.G. Madsen
DTU Transport, Technical University of Denmark

Bjørn Petersen
DTU Transport, Technical University of Denmark

Stefan Røpke
DTU Transport, Technical University of Denmark

Abstract

Passengers using public transport systems often experience waiting times when transferring between two scheduled services. In this paper we propose a planning approach which seeks to obtain a favourable trade-off between the two contrasting objectives passenger service and operating cost by modifying the timetable. The planning approach is referred to as the Simultaneous Vehicle Scheduling and Passenger Service Problem (SVSPSP). The SVSPSP is modelled as an integer programming problem, and solved using a large neighborhood search (LNS) metaheuristic. The proposed framework is tested on data inspired by the express-bus network in the Greater Copenhagen Area. The results are encouraging and indicate a potential decrease of passenger waiting times in the network of 10–20%, with the vehicle scheduling costs remaining unaffected.

In revision.

1 Introduction

In every larger public transport system massive amounts of time are wasted due to waiting time when transferring between different parts of the journey. For the Greater Copenhagen area it has been estimated that the time lost on an average weekday by passengers waiting for connecting buses or trains approaches 65,000 hours (based on 400,000 daily transfers with an average of 10 minutes transfer waiting time.[1] Hence, generating timetables which optimise for temporal correspondences has an enormous socio-economic potential. Clearly, this could be achieved through an increase in the frequency of the trips offered in the timetable, however this would require an unacceptable increase in operating costs.

The traditional sequential framework for planning of public transport has been excellently described by Desaulniers and Hickman [8] and is sketched in Figure 1. Given the route network, the frequencies are determined to ensure demand coverage and to comply with politically determined service levels, under practical constraints such as fleet size. The timetabling process then determines the exact timings for all trips while respecting the previously determined frequencies/headways. Both of these first phases are concerned with maximising some measure of passenger service, and are carried out by the public transport service provider, who typically works by appointment by the local authorities. The timetabling phase may take schedule synchronisation and transfer times into account.

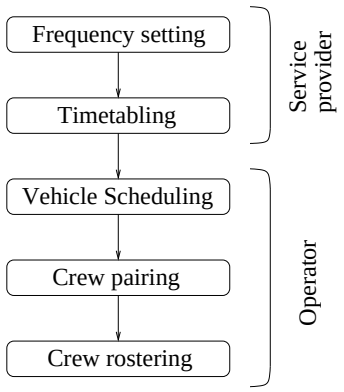


Figure 1: Traditional sequential planning approach

Once the timetable has been established, the resource scheduling starts. During this phase the first problem to be solved is the scheduling of the physical resources necessary to carry out the trips in the timetable, i.e. the vehicles. The purpose of the vehicle scheduling is to be able to execute the timetable at the lowest possible cost. The costs considered in this phase include empty mileage performed by the vehicles, both in connection to the depot, and in the form of *deadheading*, i.e. transport between the end point of one trip and the starting point of another. Once the vehicle schedules have been established, the crew pairing and rostering phases are carried out. The last three phases are all carried out by the public transport operator, who is appointed by the service provider to operate a set of trips, and they all have the purpose of operating the requested timetable at the lowest possible cost.

Today, efficient systems for generating near-optimal vehicle schedules exist within all

[1] cf. http://www.dtu.dk/centre/modelcenter/TU/Standard%20Tabeller/

modes of transport. However, these systems treat the timetable as fixed input, meaning that potential savings in operating costs from moving a set of trips in the timetable are lost. Only very limited research has been done on models that address the problem of minimising the operating costs by modifying the timetable. Furthermore, research is scarce on models that focus on minimisation of the waiting time during transfer.

In this paper we introduce the Simultaneous Vehicle Scheduling and Passenger Service Problem (SVSPSP) which addresses the multiple objective planning problem of improving timetables such that they remain economically satisfactory for the operator, and at the same time offer high-quality service to the passengers by reducing the unproductive time spent on waiting during transfers. Please note that whenever we refer to *waiting time* throughout this paper we are solely referring to the waiting time associated with transfers, and not the waiting time of passengers entering the system. The SVSPSP framework is sketched in Figure 2, and integrates the planning processes of timetabling and vehicle scheduling.

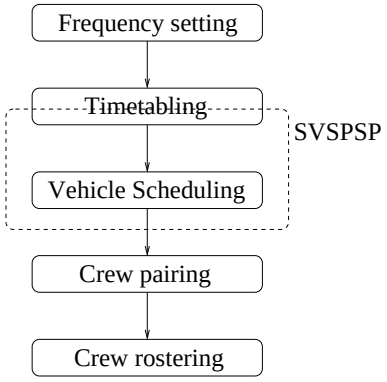


Figure 2: The role of the SVSPSP shown in relation to the traditional sequential planning approach.

Its main input is the original timetable and estimates of passenger demand in the network. The natural problem owner of the SVSPSP is the public transport service provider, as this is the authorithy which on the one hand is committed to provide a high-quality timetable to the customers (in terms of e.g. minimum waiting times) and on the other hand holds the responsibility of ensuring that the offered timetable is feasible from an operating costs perspective. By integration of the vehicle scheduling phase, which previously belonged to the operator, the service provider can obtain a better negotiating position towards the operator, since the operating costs have already been considered during the optimisation of the timetables.

The contributions of this paper are fourfold: 1) we formally introduce a new interesting problem, motivated by a real-life case, 2) we make a realistic data set available, that can be used for future studies, 3) we propose a heuristic solution method that is able to handle data sets of realistic size, 4) we show that substantial reductions in passenger waiting time are possible using the proposed methodology. The paper is organised as follows: Section 2 reviews the literature on the multiple depot vehicle scheduling problem as well as work on minimising passenger transfer times. In section 3 we formulate the SVSPSP as an integer programming model. Section 4 discusses how the proposed problem can be solved by the large neighborhood search metaheuristic. Section 5 introduces the data set used in this study which is based on the bus network of the Greater Copenhagen area, and in Section 6 we discuss

the results obtained. Finally, we provide our concluding remarks and suggest directions for further research in Section 7.

2 Literature review

Our approach for the integrated vehicle scheduling and timetabling problem is based on the *multiple depot vehicle scheduling problem* (MDVSP). Desrosiers et al. [9] provide an excellent introduction to the problem and survey the literature prior to 1995. A more recent, but short literature survey is presented by Pepin et al. [23] who also presents an interesting comparison of heuristic approaches for the problem. Section 4.1 in Desaulniers and Hickman [8] also contains a recent survey. Some of the currently best exact methods for the MDVSP are proposed by Hadjar et al. [12] and Löbel [20]. We are aware of twopapers that extend vehicle scheduling problems to handle parts of the timetabling process. The paper by van den Heuvel et al. [28] studies the integration of timetabling and multi depot vehicle scheduling with the aim of reducing costs (reducing the number of vehicles) while ignoring passenger waiting times. On the timetabling level the approach allows the trip starting times for each line to be shifted in time to allow greater flexibility in the vehicle scheduling part. The paper presents integer programming models as well as a local search algorithm that solves a network flow problem in each local search iteration. Guihaire and Hao [11] also integrate vehicle scheduling and timetable synchronisation in their optimisation problem. They consider several terms in their objective: number of vehicles required, number and quality of transfer possibilities and the so-called headway evenness. The second term aims at minimising passenger inconvenience. The last term attempts to make arrivals of vehicles, serving a particular line, occur with a regular frequency. The three terms are weighted together. In terms of the vehicle scheduling problem, the paper considers a single depot setup while our approach handles the multiple depot case. The problem studied in this paper is probably the one that resembles our problem the most.

Several papers focus on optimising timetables in order to minimise passenger waiting times, without explicitly considering the impact such changes have on the physical resource requirements (e.g. more buses may be needed to carry out the modified plan). Examples of such approaches are Jansen and Pedersen [13] who formulate the problem as a mathematical model and propose simulated annealing and tabu search algorithms to solve the problem (see also Pedersen [21]); Ceder et al. [5] who synchronise bus timetables by maximising the number of times two buses arrive at the same time at any node in the network; Klemt and Stemme [15], Bookbinder and Désilets [4] and Daduna and Voß [7] who synchronise timetables by solving a quadratic semi-assignment problem. Worth mentioning is also the paper by Chakroborty et al. [6], which studies timetable synchronisation and "optimal fleet size" using a genetic algorithm heuristic. They do not study the vehicle scheduling aspect of the problem, instead the term "optimal fleet size" refers to the fact that the number of departures on a specific line is a variable, decided by the proposed model.

As explained in the introduction, SVSPSP integrates the timetabling and vehicle scheduling phases. The integrated problem has not been widely studied in the literature but some papers on the topic do exist. One approach for handling the integrated problem has been the so-called *periodic event scheduling problem* (PESP). The PESP is mainly used for timetabling but has been extended to handle some aspects of vehicle scheduling as well. The PESP model was proposed by Serafini and Ukovich [26]. It is a general framework for modelling opti-

misation problems with a periodic nature. Liebchen and Möhring [18] show how the PESP and extensions can be used to handle many aspects of railway timetabling. One of these is to minimise the changeover time for passengers and another is the minimisation of the number of vehicles needed to perform the timetable. The complexity of the vehicle minimisation depends on whether trains are allowed to switch line when they reach their endpoint. Contrary to our approach the paper does not model the situation where vehicles can perform deadheading in order to switch terminal (this does not seem practical when the vehicles are trains running on tracks, but can be useful for buses). The material in Liebchen and Möhring [18] builds on the work of Liebchen and Peeters [19] which focuses on vehicle minimisation, but arrives at a model with a quadratic objective function. Other recent works on the PESP and railway timetabling include Liebchen and Möhring [17], Peeters [22], and Kroon et al. [16].

Wong et al. [29] studies the *Mass Transit Railway* in Hong Kong that contains 6 train lines. They minimise the overall passenger waiting time in a non-periodic fashion. The number of vehicles needed to carry out the plan is determined in advance and is kept constant. In this way it is ensured that the proposed timetable does not become too expensive to carry out, while optimising customer satisfaction. The authors present a MIP model and solve it using a heuristic that incorporates a standard MIP solver as an important component. Fleurent et al. [10] describe an optimisation system and an interactive tool for minimising passenger waiting time while keeping vehicle costs under control. The suggested approach is tested on a case from the city of Montreal, Canada, and the results indicate that the passenger waiting time can be improved while keeping the vehicle count constant. The paper provides little detail about the optimisation algorithm used to obtain these results.

We can conclude that the work on integrating time tabling and vehicle scheduling is rather limited and that Guihaire and Hao [11] is the paper that presents a problem that is most similar to the SVSPSP. The SVSPSP model is, regarding some aspects, more ambitious than the model studied by Guihaire and Hao [11] as it considers a multi-depot setting which is not the case in the aforementioned paper.

3 The SVSPSP: modelling

In a classical multi-depot vehicle scheduling problem (MDVSP) one has to cover a set of trips with a set of vehicles (based at several depots) while minimising costs. A *trip* has a start and end location, as well as a departure and arrival time. In a bus scheduling setting a trip corresponds to the movement from the start to the end of a bus line. A *line* is a collection of trips that have the same start and end locations but different departure and arrival times. A line also contains trips going in the opposite direction. The MDVSP can be modelled as follows (see Desrosiers et al. [9]): let $N = \{1, \ldots, n\}$ denote the set of trips and K the set of depots. With each depot $k \in K$ we associate a graph $G^k = (V^k, A^k)$ where the set of nodes is defined as $V^k = N \cup \{n + k\}$ with $n + k$ being the node representing the k^{th} depot. The set of arcs A^k is a subset of the set $V^k \times V^k$, with all infeasible arcs removed. An arc is infeasible if it forms an impossible connection between two trips; typically this is caused by timing constraints. For each depot $k \in K$ and each arc $(i, j) \in A^k$ we define an arc cost c^k_{ij} and we are given an upper bound v^k on the number of vehicles located at k. Using a binary variable x^k_{ij} for all $k \in K, (i, j) \in A^k$, having value 1 if and only if a vehicle from depot k

travels from node i to j we can write an integer multi-commodity flow model as follows:

$$\min \sum_{k \in K} \sum_{(i,j) \in A^k} c_{ij}^k x_{ij}^k \tag{1}$$

subject to

$$\sum_{k \in K} \sum_{j \in V^k} x_{ij}^k = 1 \qquad i \in N \tag{2}$$

$$\sum_{j \in N} x_{n+k,j}^k \leq v^k \qquad k \in K \tag{3}$$

$$\sum_{i \in V^k} x_{ij}^k - \sum_{i \in V^k} x_{ji}^k = 0 \qquad k \in K, j \in V^k \tag{4}$$

$$x_{ij}^k \in \{0,1\} \quad k \in K, (i,j) \in A^k \tag{5}$$

The objective (1) minimises the total cost. The arc costs c_{ij}^k can be set such that the total cost reflects a fixed cost per vehicle and deadheading costs. Constraints (2) ensure that all trips are served, constraints (3) ensure that we do not use more than the available number of vehicles and, constraints (4) are flow conservation constraints.

The SVSPSP generalises the MDVSP as follows: in the SVSPSP we group trips into so called *metatrips*. The set of metatrips, Ω, forms a partitioning of the set N, that is, $\cup_{M \in \Omega} M = N$ and $\forall M_1, M_2 \in \Omega, M_1 \neq M_2 : M_1 \cap M_2 = \emptyset$. Furthermore, we relax the condition that every trip must be covered. Instead we require that exactly one trip from each metatrip must be covered. In the context of this paper, we assume that each metatrip corresponds to a trip from the original timetable, and the (sub)trips belonging to the metatrip represent copies of the original trip, with alternative departure times. Thus, the requirement that each metatrip is covered corresponds to the MDVSP-requirement that each trip is covered (2). The idea behind this, in relation to our goal of increasing passenger service, is that selecting alternative departure times may reduce waiting times and thereby improve the passenger service level.

We will now introduce some useful concepts that will be used in our treatment of the SVSPSP. Trips in the SVSPSP model can be *incompatible* for various reasons, as we shall see later. This is captured by a set $\Phi \subseteq 2^N$ containing sets of mutually incompatible trips. Thus, if $\phi \in \Phi$ then any pair $i, j \in \phi$ is incompatible and cannot be used together in a feasible solution. For the SVSPSP we maintain the definition of a *line* that is known from the MDVSP; a line L is a sequence of *stops* to be visited in a given order. A line can be travelled in both directions, and we use the term d-line (directed line) for a line in a particular direction. Each metatrip, and the trips contained in it, belongs to exactly one d-line. Therefore we can view a d-line L as a subset of the set of metatrips: $L \subseteq \Omega$. For every bus line a number a *stops* are defined. The stops are the locations where the bus stops to pick up and unload passengers. Several bus lines may share one stop and a stop can provide connection to other modes of timetabled transportation like trains or ferries. Any transfer of passengers takes place at a stop. We are only interested in stops where transfers can take place, hence, when mentioning stops in the rest of this paper we assume a stop with at least one transfer opportunity.

Figure 3 shows an example of trips and metatrips. The nodes $\{1, \ldots, 12\}$ represent trips, and two metatrips $\{2, \ldots, 6\}$ and $\{7, \ldots, 11\}$ are shown. The time of day is shown along the top of the figure. Trips 4 and 9, marked with grey, are the two original trips, from which the

metatrips are constructed. The remaining trips in each metatrip are constructed by creating duplicates of the original trip, spread evenly in the available time interval. The nodes 1 and 12 belong to other metatrips, not illustrated in the figure. All trips shown in the figure belong to the same d-line.

The usage of incompatible trips to impose passenger service is apparent: trips belonging to the same d-line and departing within a short time interval should be incompatible, for example trip 6 and 7 on Figure 3 could be incompatible because they depart within 4 minutes. Similarly, two consecutive departures on a d-line should not be too far apart. Therefore it would make sense to make trip 2 incompatible with trip 11. If departures at regular intervals are required on a bus line for a specific period of the day or the entire day this could also be modelled using incompatible trips. If we desire departures every 20 minutes in the example on Figure 3 we must make trip 2 incompatible with trips 8, 9, 10, and 11 (by adding the set $\{2, 8, 9, 10, 11\}$ to Φ), trip 3 should be incompatible with trips 7, 9, 10, and 11, and so on.

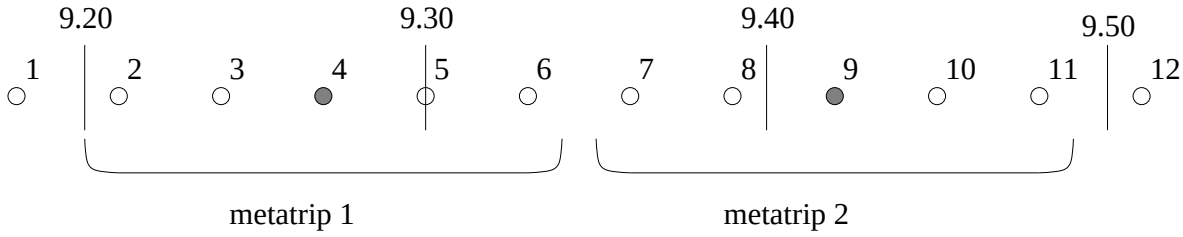


Figure 3: Example of trips and metatrips.

Using the notation from the MDVSP we can now present a mathematical model for a simple version of the SVSPSP, denoted SVSPSP^0.

$$\min \sum_{k \in K} \sum_{(i,j) \in A^k} c_{ij}^k x_{ij}^k \tag{6}$$

subject to

$$\sum_{i \in M} \sum_{k \in K} \sum_{j \in V^k} x_{ij}^k = 1 \qquad M \in \Omega \tag{7}$$

$$\sum_{i \in \phi} \sum_{k \in K} \sum_{j \in V^k} x_{ij}^k \leq 1 \qquad \phi \in \Phi \tag{8}$$

$$\sum_{j \in N} x_{n+k,j}^k \leq v^k \qquad k \in K \tag{9}$$

$$\sum_{i \in V^k} x_{ij}^k - \sum_{i \in V^k} x_{ji}^k = 0 \qquad k \in K, j \in V^k \tag{10}$$

$$x_{ij}^k \in \{0, 1\} \quad k \in K, (i,j) \in A^k \tag{11}$$

Constraints (9) and (10) are identical to (3) and (4) in the original MDVSP formulation. Constraints (7) ensure that exactly one trip from each metatrip is selected and constraints (8) ensure that no incompatible trips are selected at the same time.

In order to discuss how passenger service can be taken into account in the SVSPSP^0 we need to define exactly how we measure passenger service. The area we focus on in relation to passenger service is waiting time during transfers. We first introduce the central concept

transfer opportunity. A transfer opportunity is a triple (s, M, L). Here s is the stop where the transfer takes place, M is a metatrip that stops at s, and L is a connecting line that exchanges passengers with M at s. For each transfer opportunity we assume that an estimate D^s_{ML} of the number of passengers disembarking metatrip M and transferring to line L at stop s, as well as an estimate E^s_{ML} of the number of passengers embarking metatrip M transferring from line L at stop s are available. It is assumed that all passengers disembarking a metatrip to transfer to line L take the earliest possible departure on line L and all passengers embarking a metatrip M come from the latest possible arrival on line L. For the SVSPSP^0, L is a line external to the model, but we will later generalise it to include those lines that are rescheduled by the model.

To improve passenger service we desire to minimise the total number of passenger minutes wasted by waiting for a connection, at the same time as we want to minimise the cost of serving all trips. This results in two goals that are weighted together in the cost coefficients of the objective function. The SVSPSP^0 model can accommodate a part of the waiting times that we desire to include in the model, namely a penalty for waiting times related to lines that are external to the model, such as already timetabled train departures: for each trip i in N we find the transfer opportunities (s, M, L) of the metatrip M that i belongs to. As stated above, L is an external line with fixed departures and arrivals, therefore we can a priori find the arrival and departure on line L that are used by passengers embarking and disembarking trip i at stop s and we can calculate the associated waiting times. The two waiting times are multiplied by the passenger estimates E^s_{SM} and D^s_{SM} and summed to give the total number of minutes waited for the particular trip and transfer opportunity. By summing over all the transfer opportunities that the trip is involved in we obtain the total number of waiting minutes incurred by the trip. This number, weighted in a suitable way, is added to the cost of all arcs leaving the node corresponding to the trip.

The SVSPSP^0 model cannot take the transfer of passengers from bus to bus into account if both buses are rescheduled by the model. We therefore introduce the model SVSPSP, that generalises SVSPSP^0 to accommodate this. The overall idea is to introduce two new sets of binary variables y^s_{ij} and z^s_{ij} that indicate if transfers between trip i and j are taking place at stop s. For each transfer opportunity (s, M, L) involving a d-line L which is timetabled by the model we create a number of variables y^s_{ij} where $i \in M$, $j \in \cup_{M' \in L} M'$. Each variable indicates if the transfer opportunity of passengers disembarking metatrip M to transfer to d-line L is realised by transferring from trip i to j. Similarly, for the same transfer opportunity, we create a number of variables z^s_{ij} where $j \in M$, $i \in \cup_{M' \in L} M'$. These variables indicate if the transfer opportunity of passengers embarking M, coming from L is realised by transferring from trip i to j. We assign a cost $\bar{c}^s_{ij} > 0$ for each y^s_{ij} variable and a cost $\hat{c}^s_{ij} > 0$ for each z^s_{ij} variable. The cost is based on the time between arrival and departure on the two trips and the number of passengers expected to take advantage of the transfer opportunity.

Consider the following example: the bus lines 200 and 300 both visit *Lyngby Station*. Assume that a trip for line 200 northbound (200-N) has been chosen by the model such that the bus arrives at Lyngby station at 9:29. A number of the passengers on board the bus wish to disembark the bus to transfer to line 300 heading north (300-N). Their waiting time depends on the departure time of the next 300-N, which is also decided by the model. Figure 4 shows this situation. The chosen trip for bus 200-N (trip a) is shown at the top of the figure along with alternative 200-N arrivals and nine trips belonging to line 300-N are shown on the bottom. Passengers from trip `a` cannot transfer to bus 300-N on the departure times marked with grey circles: departure 4 is impossible because it departs before bus 200-N arrives, while

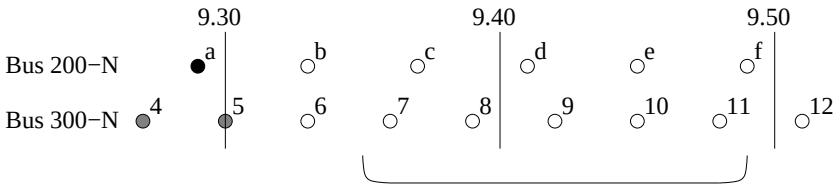


Figure 4: Example of a bus-to-bus transfer.

departure 5 departs one minute later than trip a arrives and there is not enough time for the transfer (passengers have to walk). The other departures are all feasible transfers. Note that trips 7 to 11 constitute a metatrip, so exactly one of these trips must be selected. This means that no passenger from trip a heading for line 300-N would transfer to trip 12 because an earlier, feasible departure will exist in the plan. On the other hand, if trip 12 is selected by the model and trip a is the latest selected bus from 200-N that allows a transfer to trip 12 then embarking passengers on trip 12 arriving from 200-N would perform the transfer. Since both embarking and disembarking passengers are considered, both y and z variables are necessary. The y variables handle passengers *disembarking* a specific trip to the first possible trip on the specified d-line. The z variables handle passengers *embarking* a specific trip from the last possible trip on the specified d-line.

Let S be the set of all stops that are visited by more than one bus line. We introduce a graph $\hat{G}^s = (\hat{V}^s, \hat{A}^s)$ for each stop $s \in S$. The set of vertices $\hat{V}^s$ is the set of all trips that visit stop s and the set of arcs is defined as

$$\hat{A}^s = \left\{(i,j) : i,j \in \hat{V}^s, \text{ passengers can transfer from trip } i \text{ to trip } j \text{ at stop } s\right\}.$$

For example, if s is Lyngby station as shown in Figure 4 we would have that

$$\{(a,6),(a,7),(a,8),(a,9),(a,10),(a,11),(a,12)\} \subset \hat{A}^s$$

but $\{(b,1),(b,2)\} \cap \hat{A}^s = \emptyset$. The variables y^s_{ij} and z^s_{ij} are defined for every $s \in S$ and every arc $(i,j) \in \hat{A}^s$. We can use Figure 4 to show the meaning of the y variables. If, for example, trips b and 7 are chosen and none of the trips $\{3,4,5,6\}$ are chosen then $y^s_{b,7} = 1$ and $y^s_{b,j} = 0$ for $j \in \{3,4,5,6,8,9,10\}$. If both trip 3 and 7 were chosen then we would have $y^s_{b,3} = 1$ and $y^s_{b,7} = 0$ because all passengers disembarking b, bound for 300-N, would transfer to trip 3.

For a trip $i \in N$ and a stop s on its line we define $t(i,s)$ to be the departure time of trip i at stop s. For a trip i we define $dl(i)$ to be the d-line that the trip belongs to. For a stop s and an arc $(i,j) \in \hat{A}^s$ we define

$$\pi(i,j,s) = \{j' \in \cup_{M' \in dl(j)} M' : (i,j') \in \hat{A}^s, t(j') < t(j)\},$$

that is, $\pi(i,j,s)$ is the set of trips j' from the same d-line as j that are earlier than j but that still are feasible transfer destinations from trip i. Similarly we define

$$\sigma(i,j,s) = \{i' \in \cup_{M' \in dl(i)} M' : (i',j) \in \hat{A}^s, t(i) < t(i')\},$$

which is the set of trips i' from the same d-line as i that are later than i but where a transfer to trip j still is feasible. We can now present an extended model that also handles the bus-to-bus transfers:

$$\min \sum_{k \in K} \sum_{(i,j) \in A^k} c^k_{ij} x^k_{ij} + \sum_{s \in S} \sum_{(i,j) \in \hat{A}^s} \bar{c}^s_{ij} y^s_{ij} + \sum_{s \in S} \sum_{(i,j) \in \hat{A}^s} \hat{c}^s_{ij} z^s_{ij} \tag{12}$$

subject to

$$\sum_{i\in M}\sum_{k\in K}\sum_{j\in V^k} x_{ij}^k = 1 \qquad M \in \Omega \tag{13}$$

$$\sum_{i\in \phi}\sum_{k\in K}\sum_{j\in V^k} x_{ij}^k \leq 1 \qquad \phi \in \Phi \tag{14}$$

$$\sum_{j\in N} x_{n+k,j}^k \leq v^k \qquad k \in K \tag{15}$$

$$\sum_{i\in V^k} x_{ij}^k - \sum_{i\in V^k} x_{ji}^k = 0 \qquad k \in K, j \in V^k \tag{16}$$

$$\sum_{k\in K}\sum_{l\in V^k} x_{il}^k + \sum_{k\in K}\sum_{l\in V^k} x_{jl}^k - 1 - \sum_{j'\in\pi(i,j,s)}\sum_{k\in K}\sum_{l\in V^k} x_{j'l}^k \leq y_{ij}^s \qquad s \in S, (i,j) \in \hat{A}^s \tag{17}$$

$$\sum_{k\in K}\sum_{l\in V^k} x_{il}^k + \sum_{k\in K}\sum_{l\in V^k} x_{jl}^k - 1 - \sum_{i'\in\sigma(i,j,s)}\sum_{k\in K}\sum_{l\in V^k} x_{i'l}^k \leq z_{ij}^s \qquad s \in S, (i,j) \in \hat{A}^s \tag{18}$$

$$x_{ij}^k \in \{0,1\} \quad k \in K, (i,j) \in A^k \tag{19}$$

$$y_{ij}^s \in \{0,1\} \quad s \in S, (i,j) \in \hat{A}^s \tag{20}$$

$$z_{ij}^s \in \{0,1\} \quad s \in S, (i,j) \in \hat{A}^s \tag{21}$$

Two changes have been performed compared to model SVSPSP^0: a) two terms have been added to the objective function (12) to model the cost of passengers waiting during transfers between two buses that are both re-timetabled by the model, and b) inequalities (17) and (18) have been added to ensure that the y_{ij}^s and z_{ij}^s variables are set correctly. For example, y_{ij}^s is set to 1 by (17) when both trip i and trip j are used (the first two sums on the left hand side) and when none of the feasible transfer destinations earlier than j are in use (the last sum on the left hand side). The constraints only enforce a lower bound on y_{ij}^s but the minimisation in the objective and assumption that $\bar{c}_{ij}^s$ is positive ensure that the y variables take the lowest possible value. Constraints (18) are similar to (17), but work on z rather than y variables.

The mathematical model presented in (6)–(11) has been implemented in CPLEX, but CPLEX was not able to solve instances with the dimensions considered in this paper. No attempts have been made to solve the model presented in (12)–(21) with a general purpose solver, since the number of variables and constraints used in the advanced model is even larger than in the model presented in (6)–(11). However, by presenting the models here, they have served as an instrument to give a precise definition of the problem to be studied. Using techniques like reformulation or cut or column generation it might be possible to solve realistically sized instances using the mathematical models — in particular, model (6)–(11) lends itself to a column based solution approach. However, we have worked in a different direction, and in the following section we shall present a metaheuristic for solving the problem.

4 Solution method

The solution method we propose for solving the SVSPSP is based on the large neighborhood search (LNS) metaheuristic. The LNS was proposed by Shaw [27]. As many other metaheuristics, the LNS is based on the idea of finding improving solutions in the neighborhood of an existing solution. What sets the LNS apart from other metaheuristics is that the neighborhood searched (or sampled) in the LNS is huge.

The term LNS is often confused with the term *very large scale neighborhood search* (VLSN) defined in Ahuja et al. [1]. While the LNS is a heuristic framework, VLSN is the family of heuristics that searches neighborhoods whose sizes grow exponentially as a function of the problem size, or neighborhoods that simply are too large to be searched explicitly in practice, according to Ahuja et al. [1]. The LNS is one example of a VLSN heuristic.

We are aware of one application of LNS to the MDVSP, this approach is described in Pepin et al. [23]. That LNS implementation is more complex than ours as it uses column generation and branch and bound to solve restricted instances of the MDVSP. The computational results reported in Pepin et al. [23] show that the LNS is competitive against 4 other heuristics. LNS has also been successful in solving the related vehicle routing problem with time windows. See for example Bent and van Hentenryck [3] and Pisinger and Ropke [25].

4.1 Large neighborhood search

A LNS heuristic moves from the current solution to a new, hopefully better, solution by first *destroying* the current solution and then *repairing* the destroyed solution. To illustrate this, consider the traveling salesman problem (TSP). In the TSP we are given n cities and a cost matrix that specifies the cost of traveling between each pair of cities. The goal of the TSP is to construct a minimum cost cycle that visits all cities exactly once (see e.g. Applegate et al. [2]). A destroy method for the TSP could be to remove 10% of the cities in the current tour at random (shortcutting the tour where cities are removed). The repair method could insert the removed cities again using a cheapest insertion principle (see e.g. Jünger et al. [14]).

The LNS heuristic is outlined on Algorithm 1. In the pseudo-code we use the symbols x for the current solution, x^* for the best solution observed during the search and x' for a temporary solution. The operator $d(\cdot)$ is the destroy method. When applied to a solution x it returns a partially destroyed solution. The operator $r(\cdot)$ is the repair method. It can be applied to a partially destroyed solution and returns a normal solution. The expression $r(d(x))$ therefore returns a solution created by first destroying x and then rebuilding it.

The LNS heuristic takes an initial solution as input and makes it the current and best known solutions in lines 1 and 2. Lines 4 to 10 form the main body of the heuristic. In line 4 the current solution is first destroyed and then repaired, resulting in a new solution x'. In line 5 the new solution is evaluated to see if it should replace the current solution, this is done using the function *accept* which is described in Section 4.2.3 below. In lines 8 to 10 the best known solution is updated if necessary. Line 11 checks the stopping criterion which in our implementation simply amounts to checking if $t_{\max}$ seconds have elapsed.

4.2 Large neighborhood search applied to the SVSPSP

This section describes how the LNS heuristic has been tailored to solve the SVSPSP. In particular, we describe the implemented destroy and repair methods and the acceptance

Algorithm 1 Large Neighborhood Search

```
1: input: a feasible solution x;
2: x* = x;
3: repeat
4:    x' = r(d(x));
5:    if accept(x', x) then
6:       x = x';
7:    end if
8:    if f(x') < f(x*) then
9:       x* = x';
10:   end if
11: until stop criterion is met
12: return x*
```

criterion.

4.2.1 Destroy methods

Destroy methods for the SVSPSP remove trips from the current solution. Every time a destroy method is invoked the number of trips to remove is selected randomly in the interval $[5, 30]$. Two simple destroy methods for the SVSPSP have been implemented. The first method simply remove trips at random, which is a good method for diversifying the search.

The second method is based on the *relatedness* principle proposed by Shaw [27]. Here we assign a relatedness measure $R(i, j)$ to each pair of trips (i, j). A high relatedness measure indicates that the two trips are highly related. The relatedness of two trips i and j are defined as

$$R(i,j) = 30 \times \mathbf{1}_{s(i)=s(j)} \; + \; 30 \times \mathbf{1}_{e(i)=e(j)} \; + \; 20 \times \mathbf{1}_{s(i)=e(j)} \; + \; 20 \times \mathbf{1}_{e(i)=s(j)} \; - \; |t(i) - t(j)|$$

where $s(i)$ and $e(i)$ are the start and end locations of trip i respectively, $t(i)$ is the start time of trip i (start time in the current solution). The notation $\mathbf{1}_{expr}$ is used to represent the indicator function which evaluates to one if *expr* evaluates to true and zero otherwise. The measure defines two trips to be related if they start around the same time and if the share start and/or end locations. The measure is used to remove trips as follows. An initial seed trip is selected at random and added to a set of removed trips S. For each trip i still in the solution we calculate the relatedness

$$v(i, S) = \max_{j \in S} \{R(i, j)\}$$

The trips still in the solution are sorted according a non-increasing $v(i, S)$ in a sequence T, a random number p in the interval $[0, 1)$ is drawn and the trip at position $\lfloor |T| p^5 \rfloor$ in T is selected. This selection rule favours trips with high $v(i, S)$ value. The selected trip is added to the set of removed trips, and $v(i, S)$ is recalculated after adding a trip to S. We continue to add trips to S, until we have reached the target number of removed trips.

The two destroy methods are mixed in the LNS heuristic. Before removing a trip from the solution it is decided which destroy method that should be used to select the trip. With probability 0.15 the first method (random) is used and with probability 0.85 the second method (relatedness) is used.

The trips that have been removed from the solution are still *active* in the sense that they will be used in the trip incompatibility check defined by constraints (8) and (14). That is when adding a trip to a solution in the repair step below, we check if it is compatible with the trips in the solution and the trips removed in the previous destroy operation. A trip i is made inactive when another trip, belonging to the same metatrip as i, is inserted into the solution.

4.2.2 Repair methods

The repair method for the SVSPSP reinserts the trips that were removed from the solution by the destroy method. The repair method uses a randomised greedy heuristic. For each unassigned metatrip S the heuristic calculates an insertion cost $f(S)$ given the current solution. When inserting a metatrip S we have a choice of which trip $i \in S$ that should be inserted. With probability ρ we insert the same trip that was used in the solution before destruction and with probability $1 - \rho$ we insert a random trip from S. The chosen trip should be compatible with all active trips. Such a trip exists because we are sure that the trip from the pre-destruction solution is compatible with all trips. The requirement ensures that we never get to a situation where one or more metatrips cannot be inserted because of the the compatibility constraints (8) and (14).

Given the choice of trip i, we define the cost $f(S)$ as the cost of inserting trip i at the best possible position in the current solution multiplied by a random factor that is meant to diversify the insertion procedure. More precisely the cost is defined as:

$$f(S) = \begin{cases} \min_{r \in R}\{c(i,r)\} \cdot (1 + \text{rand}(-\delta, \delta)) & \text{if } \min_{r \in R} c(i,r) \neq \infty \\ c(i, \emptyset) & \text{otherwise} \end{cases}$$

where $c(i,r)$ is the cost of inserting trip i in route r at the best possible position, R is the set of routes in the current solution, $c(i,\emptyset)$ is the cost of serving the trip using a new vehicle from the best possible depot, δ is a parameter and $\text{rand}(-\delta,\delta)$ is a function that returns a random number in the interval $[-\delta,\delta]$. The parameter δ controls the amount of randomisation applied by the insertion procedure. The heuristic chooses to insert the metatrip S with lowest cost. It does this by inserting the trip i that was used as a representative for S and inserts this at its best possible position. This continues until all metatrips have been inserted. With to the assumption that $v^k = |\Omega|$ it is always possible to insert a metatrip — we will always be able to serve it using a new vehicle.

4.2.3 Acceptance criterion

The acceptance criterion used in our implementation of the LNS heuristic is the one used in simulated annealing metaheuristics: The function $\text{accept}(x', x)$ used in line 6 of Algorithm 1 accepts the new solution x' if it is at least as good as the current solution x, that is, $f(x') \leq f(x)$. If $f(x') > f(x)$ then the solution is accepted with probability

$$e^{\frac{f(x)-f(x')}{T}}.$$

The parameter T is called the *temperature* and controls the acceptance probability: a high temperature makes it more likely that worse solutions are accepted. Normally the temperature is reduced in every iteration using the formula $T^{\text{new}} = \alpha T^{\text{old}}$ where $0 < \alpha < 1$ is a parameter

Algorithm 2 Heuristic for generating an initial solution

```
1: while there are non-served metatrips left do
2:   Select a random station s with unserved metatrips;
3:   Select earliest non-served metatrip S starting from s;
4:   Start a new route r serving S. Use a vehicle from the depot nearest to s;
5:   repeat
6:     Let s' be the station where route r is ending;
7:     if r can be extended with a non-served metatrip starting in s' then
8:       Select earliest non-served metatrip S' starting in s' that can extend r. Add S' to
         r;
9:     else
10:      End route r by returning to the depot;
11:    end if
12:  until r has returned to the depot;
13: end while;
```

that is set relative to desired start and end temperatures and desired number of iterations. Because we use elapsed time as stopping criterion we calculate the current temperature by the formula

$$T(t) = T_s \cdot \left(\frac{T_e}{T_s}\right)^{\frac{t}{t_{\max}}}$$

here t is the elapsed time since the start of the heuristic, T_s is the starting temperature and T_e is the end temperature. Because of the acceptance criterion the LNS heuristic can be seen as a simulated annealing heuristic with a complex neighborhood definition.

4.2.4 Starting solution

A starting solution is necessary because the LNS heuristic improves an existing solution. It is constructed using the greedy heuristic outlined in Algorithm 2. The generation heuristic does not consider time shifting, instead it only considers insertion of the original trip from each metatrip. Therefore, when writing *earliest metatrip* in Algorithm 2 we refer to the metatrip whose original trip is the earliest. The heuristic constructs vehicle routes one at a time and attempts to create routes where little time is wasted in between trips. Lines 2–12 deal with the construction of a single route for a vehicle. Lines 2–4 select the first trip on the route and the depot which should provide the vehicle for the route. Lines 5–12 add trips to the partial route. The selection of which trip to add is based on the terminal where the partial route is ending at the moment. The algorithm adds the first trip that leaves that terminal or closes the route if the route cannot be extended with a trip starting in the current terminal.

5 Data

The data set that has been developed for the SVSPSP during the preparation of this paper has been described in further detail in a technical report by Petersen et al. [24], and in this section we will give a brief description of the background and the resulting data set. The data set can be obtained from http://www.transport.dtu.dk/SVSPSP/.

The local train network in the Greater Copenhagen area roughly has the form of a fan or the fingers of a hand, as shown in Figure 5. A network of express bus lines complements the train lines across and in parallel, as can be seen in Figure 6. The data set that has been developed for the SVSPSP is based on this structure, where the radial train lines are operated on a fixed timetable, and the timetables for the bus lines (of which most are circular) are adjusted according to this.

Figure 5: The local train network of Copenhagen

A data set for the SVSPSP consists of several parts: 1) a *distance matrix*, containing all distances between depots and line end-points, 2) *fixed time tables* of all fixed-schedule train connections, 3) *number of transferring passengers* for each transfer opportunity, 4) *an initial schedule* used to determine the available set of trips, 5) *costs* of different activities, and other parameters such as turnaround times, passenger transfer times, etc.

Among these elements the distances and fixed time tables are generally relatively easy to obtain. Furthermore the initial schedule, in the form of the current bus schedule, is required to provide information regarding frequencies and service level, which will be maintained by the new solution. Given a suitable generation strategy, the set of potential trips can be generated based on these time tables.The current schedule can also be used to generate an initial feasible (VSP) solution for the heuristics.

The problem objectives of operating cost and passenger waiting time have been combined by expressing both in monetary units. The various costs required for calculating the total cost of a solution have been estimated for the data set, in particular the cost of passenger waiting time has been estimated based on the recommended value of travel time by the Danish

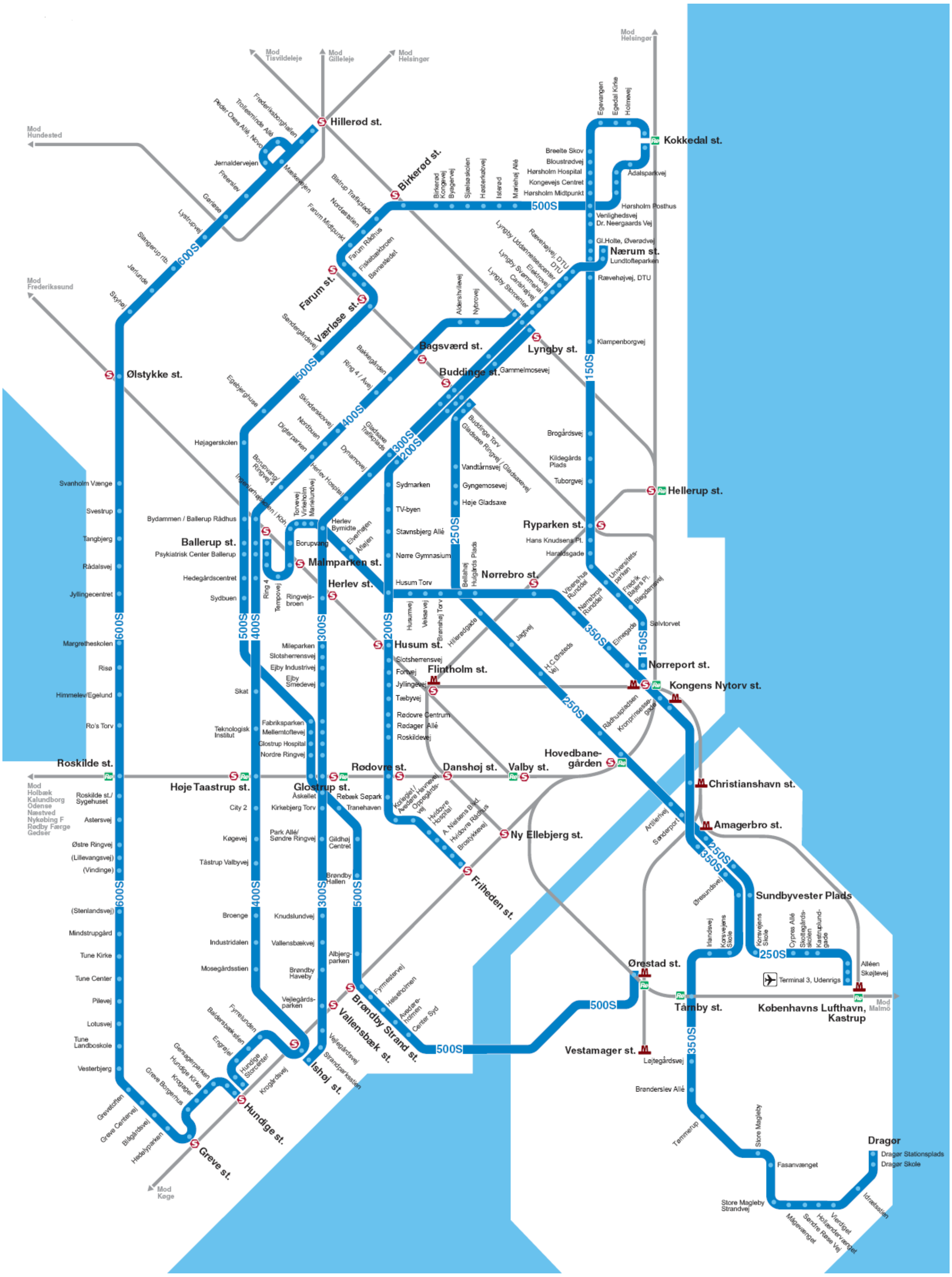


Figure 6: The S-bus network; trains are shown as thin lines, compare Figure 5

Ministry of Transport.

What then remains to be estimated is the number of passengers and their transfer patterns. This transfer information will allow us to calculate the number of (dis)embarking passengers using each available transfer opportunity, for any arrival or departure of a bus at a station.

For this project these data have been obtained by a two-stage process: First we estimated the number of (dis)embarking passengers, as a function of the station, bus line and time of day, and then we estimated the percentage of (dis)embarking passengers that sould perform each possible transfer.

The number of (dis)embarking passengers at each station is calculated as $f_t \cdot f_l \cdot f_s \cdot n$ where f_t is a time factor, f_l is a line factor, f_s is a station factor, and n is a random number evenly distributed in the interval $[32, 48]$. The values of n is chosen to roughly reflect the capacity of a vehicle, and the introduction of randomness increases the variation of data, to make them more realistic.

The distribution of transferring passengers between available connections has been estimated based on knowledge of the network, and considering the direction of trains (towards the town centre or away from it). A random element has been added to provide a better spread of the obtained values. Connections have been specified either for a particular train line or as e.g. "the first departure going into town". For modelling purposes this could be obtained by adding artificial train lines.

Metatrips are created from trips in the original timetable. Let T_i be the departure time of a trip in the original timetable, belonging to a particular d-line L. We create an interval $[T_i^s, T_i^e]$ around T_i and distribute κ trips in this interval to form a metatrip. Assume that κ is an uneven number. We express the start and end of the interval as follows $T_i^s = T_i - \tau_i^-$ and $T_i^e = T_i + \tau_i^+$. The symbols τ_i^- and τ_i^+ are expressed in terms terms of the departure times T_{i-1} and T_{i+1} of the previous and next, respectively, trip on L as follows: $\tau_i^- = \lfloor \frac{T_i - T_{i-1} - 1}{2} \rfloor, \tau_i^+ = \lfloor \frac{T_{i+1} - T_i}{2} \rfloor$. This construction ensures that the intervals around the trips on each d-line are disjoint. The set of departure times constructed are

$$\left\{ T_i - \frac{2j}{\kappa}\tau_i^- : j = 1 \dots \left\lfloor \frac{\kappa}{2} \right\rfloor \right\} \cup \{T_i\} \cup \left\{ T_i + \frac{2j}{\kappa}\tau_i^+ : j = 1 \dots \left\lfloor \frac{\kappa}{2} \right\rfloor \right\}$$

with the time expressions rounded to the nearest integer to ensure that departures occur at integer valued times. If the trips in the original timetable are close then we may end up with fewer than κ departure times because some departures get mapped to the same integer due to the round-off. In that case we only create as many trips as we have departure times for. In our test we used $\kappa = 5$. Figure 7 shows an example of how the trips of a metatrip are distributed.

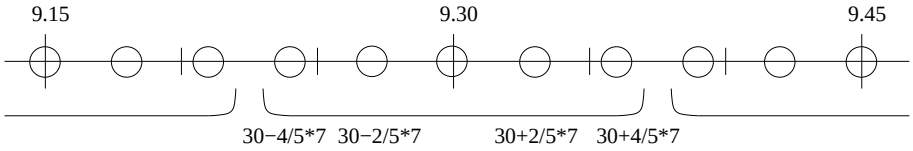


Figure 7: Example of the distribution of trips in metatrips

The only incompatibilities used in this project are found by multiplying the current interval between two trips by a factor to determine lower and upper bounds allowed for the same interval. This factor has been set to 0.5 for the lower bound and 1.5 for the upper bound.

Instances of three different sizes have been considered for this project. These instances have been constructed by considering a meaningful subset of the actual operated bus routes, i.e. a subset that in itself constitutes a realistic problem. This means that the routes selected for the smaller subset have characteristics that may differ from the routes added in the larger subsets. Thus the smaller problem consists of the most central lines, and the lines that are added in the larger sets are more rural, and/or have fewer intersections with the train network.

The properties of the three different instances will be summarised below:

3 lines. 538 trips. All lines are circular lines with 5–6 intersections with the train network, but only few interconnections between the buses. Many passengers. Subset of

5 lines. 792 trips. All lines are circular lines with 4–6 intersections with the train network, and only few interconnections between the buses. Some lines are passenger intensive. Subset of

8 lines. 1400 trips. Combination of circular and radial lines. The radial lines only have 2–3 connections to trains, but more connections to other buses. Most lines are passenger intensive.

6 Computational experiments

To evaluate the quality and usefulness of the algorithm, we have performed a series of tests to examine its behaviour with different instance sizes and settings, which will be presented in this section. The tests have been performed on an Intel Pentium 4, 2.8 GHz, with 2GB RAM, running Windows XP.

The current vehicle schedules used for the data set were not available, so these had to be constructed initially. This has been done by using the implemented LNS as a regular VSP solver, i.e. by not allowing any time shifts. The generated solutions have been used as initial solutions when solving the SVSPSP, and also as reference solutions representing current practive, when evaluating the quality of the obtained final solutions. As we know that the actual current schedules are not created with dedicated software, this should produce reference solutions that are not worse than the currently used solution. For each instance a running time of 24 hours was allowed for the construction of the reference solution.

Table 1 shows the results from running the implemented LNS algorithm on instances of different sizes with different running times. For each run we report the cost reduction compared to the initial solution, the number of vehicles used, the reduction of empty mileage costs (i.e. a negative value indicates that the empty cost has increased), the reduction of total passenger waiting time, the percentage of trips that have been time shifted, the average amount of time that each trip is shifted, and the percentage of trips that are *regular*. A *regular*/memorable trip is a trip for which the gap to the preceding trip on the same line is a multiple of 5. This makes the schedule easier to remember, and is thus an advantage to the passengers. For the current schedule the percentage of regular trips is around 72% for the largest instance, and 83–84% for the others. However, memorability has not been an objective of the implemented algorithm.

The table shows that good results can be obtained, and that a considerable reduction of passenger waiting time is possible. The reduced waiting times lead to an increase in the amount of empty travel, however the total operating cost still shows improvement of around 3% for the smaller instances, and 1–2% for the 8 line instances.

3 lines

	total cost	veh.	empty	time	shifts	avg. shift	reg.
1h	2.9%	0.0%	−14.2%	16.5%	74.2%	2.19	39.7%
6h	3.1%	0.0%	−13.0%	17.4%	73.4%	2.22	43.2%
24h	3.3%	0.0%	−8.9%	18.1%	73.8%	2.11	48.1%

5 lines

	total cost	veh.	empty	time	shifts	avg. shift	reg.
1h	2.8%	0.0%	−10.1%	19.8%	77.0%	2.58	39.8%
6h	3.1%	0.0%	−9.2%	21.8%	79.3%	2.68	43.4%
24h	3.2%	0.0%	−7.8%	22.5%	78.2%	2.61	40.5%

8 lines

	total cost	veh.	empty	time	shifts	avg. shift	reg.
1h	1.1%	0.0%	−7.8%	9.5%	64.2%	1.88	30.4%
6h	1.6%	0.0%	−6.4%	13.3%	76.6%	2.38	31.4%
24h	2.0%	0.0%	−7.1%	16.4%	76.4%	2.39	36.0%

Table 1: Solution improvements for different problem sizes

Alternative small instances

As stated previously the different tested instances differ not only in size, but also in some characteristics regarding the type of lines that are used. Thus the variation in cost and time reduction obtained for the different instances may well depend just as much on the change in these characteristics as on the actual size of the problems. The tests of Table 1 have been repeated on two additional small instances that have been created with a mix of lines more similar to those of the largest instance. These instances represent subproblems that would most likely not be considered in real-life, but can hopefully demonstrate the behaviour on smaller instances without being affected by the different characteristics of the problem. Each instance consists of two circular lines (of which one is passenger intensive) and one radial line. The results for these two instances can be found in Table 2, and indicate that it is difficult to compare the properties of instance just by looking at simple properties of the included lines. The results also indicate that the achievable cost improvement does indeed depend on the choice of lines to include in the problem.

	total cost	veh.	empty	time	shifts	avg. shift	reg.
1h	1.3%	0.0%	−7.2%	12.2%	73.2%	2.0	29.8%
6h	1.6%	0.0%	−7.7%	14.7%	76.4%	2.1	31.4%
1h	2.9%	0.0%	−8.6%	20.4%	79.4%	2.8	39.2%
6h	3.1%	0.0%	−5.6%	21.3%	76.5%	2.8	45.0%

Table 2: Solution improvements for more "realistic" small instances

Random variation of the instances

The network structure and the existing time tables are fixed, so in order to produce a series of different data sets/problem instances that still reflect the real world, the only adjustable parameter has been the random element of the spread of the passengers over different available connections. This has been done for the medium-sized instances (5 lines), using running times of 1 and 6 hours, and the results can be found in Table 3.

	total cost	veh.	empty	time	shifts	avg. shift	reg.
1h	2.8%	0.0%	−10.5%	19.7%	78.8%	2.7	37.3%
	2.2%	0.0%	−6.4%	15.4%	75.5%	2.5	39.3%
	2.8%	0.0%	−11.8%	20.1%	77.8%	2.7	34.5%
	2.7%	0.0%	−11.6%	19.7%	76.8%	2.7	39.6%
6h	3.2%	0.0%	−6.2%	21.8%	76.4%	2.6	39.9%
	2.6%	0.0%	−4.8%	17.8%	77.1%	2.7	43.1%
	3.1%	0.0%	−9.0%	21.8%	78.3%	2.6	43.1%
	3.2%	0.0%	−5.4%	21.8%	76.4%	2.5	39.5%

Table 3: Solution results with modified transfer distributions

These results show that the actual distribution of the passengers to some extent influences the size of the reductions that can be obtained, but also that the improvements are consistently around 2.6% for the shorter running times, and around 3% for the 6 hour running times.

7 Conclusion

We have introduced a new problem that integrates the timetabling and vehicle scheduling phases in public transportation planning. It does so by simultaneously considering resource costs and passenger waiting time at transfers. The problem has been defined formally and a metaheuristic based on the LNS principle has been designed and tested. The metaheuristic has been tested on a data set based on a subset of the buses serving the Greater Copenhagen area. The results obtained are encouraging: for the full data set we have observed that a 16% reduction of passenger transfer waiting times are possible. This reduction was possible without using more buses to provide the service, but an increase in the amount of deadheading was necessary. We consider the increase in deadheading negligible compared to the total cost involved in operating a public transport system and when considering the increased passenger service obtained.

A topic for future research is how to make the timetables produced by the heuristic easier for the passengers to memorise. This could be achieved either by adding a term penalising solutions with low memorability to the objective function or ensuring that blocks of subsequent departures have fixed headway.

Acknowledgment

This project has been supported by a grant from The Danish Social Science Research Council.

References

[1] R. K. Ahuja, Ö. Ergun, J. B. Ergun, and A. P. Punnen. A survey of very large-scale neighborhood search techniques. *Discrete Applied Mathematics*, 123:75–102, 2002.

[2] D.L. Applegate, R.E. Bixby, V. Chvátal, and W.J. Cook. *The Traveling Salesman Problem: A Computational Study.* Princeton University Press, 2006.

[3] R. Bent and P. van Hentenryck. A two-stage hybrid local search for the vehicle routing problem with time windows. *Transportation Science*, 38(4):515–530, 2004.

[4] J. H. Bookbinder and A. Désilets. Transfer optimization in a transit network. *Transportation Science*, 26(2):106–118, 1992.

[5] A. Ceder, B. Golany, and O. Tal. Creating bus timetables with maximal synchronization. *Transportation Research Part A*, 35:913–928, 2001.

[6] P. Chakroborty, K. Deb, and R. K. Sharma. Optimal fleet size distribution and scheduling of transit systems using genetic algorithms. *Transportation Planning and Technology*, 24 (3):209–225, 2001.

[7] J. R. Daduna and S. Voß. Practical experiences in schedule synchronization. In J.R. Daduna, I. Branco, and J.M.P. Paixão, editors, *Computer-Aided Transit Scheduling: Proceedings of the 6th International Workshop on Computer-aided Scheduling of Public Transport*, pages 39–55. Springer, 1995.

[8] G. Desaulniers and M. Hickman. Public transit. In G. Laporte and C. Barnhart, editors, *Transportation*, Handbooks in Operations Research and Management Science, chapter 3, pages 69–127. Elsevier, 2007.

[9] J. Desrosiers, Y. Dumas, M. M. Solomon, and F. Soumis. Time constrained routing and scheduling. In M.O. Ball, T.L. Magnanti, C.L. Monma, and G.L. Nemhauser, editors, *Network Routing*, volume 8 of *Handbooks in Operations Research and Management Science*, chapter 2, pages 35–140. Elsevier, 1995.

[10] C. Fleurent, R. Lessard, and L. Séguin. Transit timetable synchronization: Evaluation and optimization. Technical report, GIRO, 75, rue de Port-Royal Est, Bureau 500, Montreal (Quebec), 2007.

[11] V. Guihaire and J.-K. Hao. Transit network re-timetabling and vehicle scheduling. *Communications in Computer and Information Science*, 14:135–144, 2008. Forthcoming.

[12] A. Hadjar, O. Marcotte, and F. Soumis. A branch-and-cut algorihtm for the multiple depot scheduling problem. *Operations Research*, 54(1):130–149, 2006.

[13] L. N. Jansen and M. B. Pedersen. Minimering af skiftetider i køreplaner. Master's thesis, Centre for Traffic & Transport, Technical University of Denmark, March 2002. In Danish.

[14] M. Jünger, G. Reinelt, and G. Rinaldi. The traveling salesman problem. In M.O. Ball, T.L. Magnanti, C.L. Monma, and G.L. Nemhauser, editors, *Network routing*, volume 8 of *Handbooks in operations research and management science*, pages 225–330. Elsevier, 1995.

[15] W.-D. Klemt and W. Stemme. Schedule synchronization for public transit networks. In J.R. Daduna and A. Wren, editors, *Computer-Aided Transit Scheduling, Proceedings of the Fourth International Workshop on Computer-Aided Scheduling of Public Transport*, pages 327–335. Springer, 1988.

[16] L. Kroon, D. Huisman, and G. Maróti. Railway timetabling from an operations research perspective. Technical Report EI2007-22, Econometric Institute, 2007.

[17] C. Liebchen and R. H. Möhring. A case study in periodic timetabling. *Electronic Notes in Theoretical Computer Science*, 66(6):18–31, 2002.

[18] C. Liebchen and R. H. Möhring. The modeling power of the periodic event scheduling problem: Railway timetables — and beyond. In F. Geraets, editor, *Algorithmic Methods for Railway Optimization*, volume 4359 of *Lecture Notes in Computer Science*, pages 3–40. Springer, 2007.

[19] C. Liebchen and L. Peeters. Some practical aspects of periodic timetabling. In P. Chmoni, R. Leisten, A. Martin, J. Minnemann, and H. Stadtler, editors, *Operations Research 2001*, Heidelberg, 2002. Springer.

[20] A. Löbel. Solving large-scale multiple-depot vehicle scheduling problems. In N.H.M. Wilson, editor, *Computer-Aided Transit Scheduling*, pages 193–220. Springer, Berlin, 1999.

[21] M. B. Pedersen. Minimizing passenger transfer times in public transport timetables. *Orbit*, pages 13–20, 2003. Special ISMP issue of the newsletter of the Danish Operations Research Society.

[22] L. W. P. Peeters. *Cyclic Railway Timetable Optimization*. PhD thesis, Erasmus Research Institute of Management (ERIM), 2003.

[23] A.-S. Pepin, G. Desaulniers, A. Hertz, and D. Huisman. A comparison of five heuristics for the multiple depot vehicle scheduling problem. *Journal of Scheduling*, 12(1):17–30, 2009.

[24] H. L. Petersen, A. Larsen, O. B. G. Madsen, and S. Ropke. A data set for the simultaneous vehicle scheduling and passenger service problem. Technical Report 2008-8, DTU Transport, Technical University of Denmark, 2008.

[25] D. Pisinger and S. Ropke. A general heuristic for vehicle routing problems. *Computers & Operations Research*, 34(8):2403–2435, 2007.

[26] P. Serafini and W. Ukovich. A mathematical model for periodic scheduling problems. *SIAM Journal on Discrete Mathematics*, 2(4):550–581, 1989.

[27] P. Shaw. Using constraint programming and local search methods to solve vehicle routing problems. In *CP-98 (Fourth International Conference on Principles and Practice of Constraint Programming)*, volume 1520 of *Lecture Notes in Computer Science*, pages 417–431, 1998.

[28] A. P. R. van den Heuvel, J. M. van den Akker, and M. E. van Koten Niekerk. Integrating timetabling and vehicle scheduling in public bus transportation. Technical Report UU-CS-2008-003, Department of Information and Computing Sciences, Utrecht University, Utrecht, The Netherlands, February 2008.

[29] R.C.W. Wong, T.W.Y. Yuen, K.W. Fung, and J.M.Y. Leung. Optimizing timetable synchronization for rail mass transit. *Transportation Science*, 42(1):57–69, 2008.

Chapter 11

The Multi-Commodity k-splittable Maximum Flow Problem

Mette Gamst
DTU Management Engineering, Technical University of Denmark

Bjørn Petersen
DTU Management Engineering, Technical University of Denmark

Abstract

The Multi-Commodity k-splittable Maximum Flow Problem consists of routing as much flow as possible through a capacitated network such that each commodity uses at most k paths and the capacities are satisfied. The problem has previously been solved to optimality through branch-and-price. In this paper we propose two exact solution methods both based on an alternative decomposition. The two methods differ in their branching strategy. The first method, which branches on forbidden edge sequences, shows some performance difficulty due to large search trees. The second method, which branches on forbidden and forced edge sequences, demonstrates much better performance. The latter also outperforms a leading exact solution method from the literature. Furthermore, a heuristic algorithm is presented. The heuristic is fast and yields good solution values.

Keywords: Multi-Commodity flow, k-splittable, Dantzig-Wolfe decomposition, branch-and-price.

1 Introduction

The Multi-Commodity k-splittable Maximum Flow Problem (MCkMFP) consists of maximizing the amount of routed flow through a capacitated network such that each commodity uses at most k paths and the capacities are satisfied. The MCkMFP appears in the transportation sector when a number of commodities must be routed using only a limited number of transportation units, and in telecommunication for limiting the number of used network connections.

In revision.

The Multi-Commodity k-splittable Flow Problem (MCkFP) was presented by Baier et al. [1], who solved the Maximum Budget-Constrained Single- and Multi-Commodity k-splittable Flow Problems using approximation algorithms. The authors proved that the Maximum Single-Commodity k-splittable Flow Problem is $\mathcal{NP}$-hard in the strong sense for directed graphs. Finally, they noted that for $k \geq |E|$, a k-splittable (s,t) flow problem degenerates to an ordinary (s,t) flow problem.

Koch et al. [7] proved that the MCkMFP is $\mathcal{NP}$-hard in the strong sense for directed as well as undirected graphs, and showed that when $\mathcal{P} \neq \mathcal{NP}$, the best possible approximation factor is $\frac{5}{6}$. Koch et al. [6] considered the MCkMFP as a two-stage problem, where the first stage consists of the decision on the k paths (routing) and the second of the amount of flow on the paths (packing). If k is a constant then it suffices to consider a polynomial number of packing alternatives, which can be constructed in polynomial time. If k is part of the input, they proposed an approximation algorithm having approximation factor $(1-\epsilon)$, $\epsilon > 0$.

Truffot and Duhamel [8] used branch-and-price to solve the Single-Commodity k-splittable Maximum Flow Problem (SCkMFP). A 3-index edge-path model was presented to which a branch-and-price algorithm was applied. The pricing problem for the column generation is a shortest path problem solvable in polynomial time. Truffot et al. [9] have applied their 3-index branch-and-price algorithm to the MCkMFP.

Gamst et al. [5] used branch-and-price to solve the Minimum Cost Multi-Commodity k-splittable Flow Problem (MCMCkFP). They applied the algorithm of Truffot et al. [9] to the MCMCkFP. Furthermore, they proposed a new branch-and-price algorithm based on a 2-index model. The latter showed very good performance and outperformed the existing branch-and-price algorithm.

The MCkMFP can be represented by a directed graph $G = (V, E)$, where V is the set of vertices and E the set of edges. A positive capacity u_e is associated with every edge $e \in E$. Edge capacities are positive since any edge $e \in E$ with non-positive capacity can be removed from the graph. The set of commodities is denoted L and each commodity $l \in L$ has a source $s_l \in E$ and a destination $t_l \in E$. The maximal number of routes each commodity may use is denoted k.

In this paper three exact solution methods are applied to the MCkMFP and compared. The 3-index branch-and-price algorithm (3BP) by Truffot et al. [9] is extended with a heuristic proposed by Gamst et al. [5] to reach feasible solutions faster. The extended 3BP is compared to two algorithms based on a 2-index branch-and-price approach applied to the MCkMFP by Gamst et al. [5]. The two algorithms for the MCkMFP differ in their branching scheme. The first algorithm (2BP) uses a branching strategy from the literature where certain subpaths are forbidden, and the second algorithm (2BP') uses a new branching strategy where the use of certain paths is either forced or forbidden and where branch cuts are added to the master problem.

The main contribution of this paper is to apply the 2BP algorithm to the MCkMFP and especially to introduce the new branching scheme and the branch cuts of the 2BP' algorithm. Furthermore, a heuristic use of the 2BP and 2BP' algorithms is presented, denoted 2HEUR.

The paper is organized as follows. First, we summarize and combine exact methods from the literature on MCkMFP into an overall 3-index branch-and-price algorithm in Section 2. The 2BP algorithm is presented in Section 3, which is followed by the 2BP' algorithm in

Section 4. All algorithms are tested and compared in Section 5. Section 6 concludes the paper.

2 The 3-index branch-and-price algorithm (3BP)

Truffot et al. [9] solved the MCkMFP by applying Dantzig-Wolfe decomposition Dantzig and Wolfe [4]. We denote their branch-and-price algorithm 3BP. The pricing problem finds the h'th path of commodity l and the master problem merges paths into an overall feasible solution. In the master problem, the variable $x_p^{hl} \geq 0$ denotes the amount of flow on path p for the h'th path of commodity l and the binary variable y_p^{hl} denotes whether or not path p is used as the h'th path for commodity l. The 3BP problem is:

$$\max \quad \sum_{l \in L} \sum_{h=1}^{k} \sum_{p \in P^l} x_p^{hl}$$

$$\text{s.t.} \quad \sum_{l \in L} \sum_{h=1}^{k} \sum_{p \in P^l} \delta_e^p x_p^{hl} \leq u_e \quad \forall e \in E \tag{1}$$

$$x_p^{hl} - u_p y_p^{hl} \leq 0 \qquad \forall l \in L, h \in \{1, \ldots, k\}, \forall p \in P^l \tag{2}$$

$$\sum_{p \in P^l} y_p^{hl} \leq 1 \qquad \forall l \in L, h \in \{1, \ldots, k\} \tag{3}$$

$$x_p^{hl} \geq 0 \qquad \forall l \in L, h \in \{1, \ldots, k\}, \forall p \in P^l$$

$$y_p^{hl} \in \{0, 1\} \qquad \forall l \in L, h \in \{1, \ldots, k\}, \forall p \in P^l$$

The objective function maximizes the total amount of routed flow. The set P^l contains paths p for commodity l. In capacity constraints (1), δ_e^p indicates whether or not edge e is used by path p. The constant u_p denotes the capacity constraint on path p, which is defined as $u_p = min\{u_e \,|\, e \in p\}$. Hence, constraints (2) force the decision variable y_p^{hl} to be set if there is flow on the corresponding path x_p^{hl}. Constraints (3) ensure that at most one path is used as the h'th path of a commodity l. The path index $h \in \{1, \ldots, k\}$ causes symmetry in the solution space, hence a symmetry-breaking constraint is added to the formulation:

$$\sum_{p \in P^l} x_p^{h+1l} - \sum_{p \in P^l} x_p^{hl} \leq 0 \quad \forall h \in \{1, \ldots, k-1\}, \forall l \in L \tag{4}$$

The constraint eliminates some symmetry, but does not prevent symmetric solutions where paths carry the same amount of flow. The 3-index model is LP-relaxed by setting $0 \leq y_p^{hl} \leq 1$ and then the model is simplified by substituting x_p^{hl}/u_p for y_p^{hl}, which is feasible according to constraints (2) and (3) and to the fact that $u_p > 0$. Constraints (2) and the bounds on the y_p^{hl} variables are removed from the formulation and constraints (3) are rewritten as:

$$\sum_{p \in P} \frac{x_p^{hl}}{u_p} \leq 1 \quad \forall l \in L, h \in \{1, \ldots, k\} \tag{5}$$

Gamst et al. [5] applied the 3BP algorithm to The Minimum Cost k-splittable Flow Problem and argued that the path index $h \in \{1, \ldots, k\}$ still causes symmetry in the solution

space as well as a large number of columns in the master problem. They improved the 3BP algorithm by including a heuristic, which merges certain fractional columns such that a feasible solution was possibly reached. Specifically, one of the following two situations may occur:

1. For a commodity, several identical paths are used but with different values of h
2. More than one path is used for a single value of h for a commodity

In the first case, the paths are merged into one single path. In the second case, each path is assigned a unique value of h, if possible.

Adding this heuristic to the 3-index branch-and-price algorithm gives us the final 3BP algorithm. We do not expect the heuristic to solve all symmetry problems caused by the path index, hence a branch-and-price algorithm (2BP) without the path index for The Minimum Cost k-splittable Flow Problem by Gamst et al. [5] is investigated. In the following sections we show that the 2BP algorithm can be applied to the MCkMFP, and we present a branch-and-price algorithm (2BP') based on the same master problem as in the 2BP algorithm, but with a different branching strategy.

3 The 2-index branch-and-price algorithm (2BP)

Applying Dantzig-Wolfe decomposition to the edge-based model without using the h-index gives a pricing problem, which generates a path for each commodity, and a master problem, which merges the paths into an overall feasible solution. Let $x_p^l \geq 0$ denote the amount of flow on path p for commodity l and let $y_p^l \in \{0,1\}$ denote whether or not path p is used by commodity l. The master problem is:

$$\begin{aligned}
\max \quad & \sum_{l \in L} \sum_{p \in P^l} x_p^l & & \\
\text{s.t.} \quad & \sum_{l \in L} \sum_{p \in P^l} \delta_e^p x_p^l \leq u_e & \forall e \in E & (6)\\
& x_p^l - u_p y_p^l \leq 0 & \forall l \in L, \forall p \in P^l & (7)\\
& \sum_{p \in P^l} y_p^l \leq k & \forall l \in L & (8)\\
& x_p^l \geq 0 & \forall l \in L, \forall p \in P^l & \\
& y_p^l \in \{0,1\} & \forall l \in L, \forall p \in P^l &
\end{aligned}$$

The objective function maximizes the total amount of routed flow. Constraints (6) ensure edge capacities are never violated and constraints (7) force the decision variables to take on value 1, whenever the amount of flow on the corresponding path is positive. Constraints (8) limit the number of used paths for commodity l to at most k.

By LP-relaxing the binary variables y_p^l to $0 \leq y_p^l \leq 1$ the model is turned into an LP-model. Setting $y_p^l = x_p^l / u_p$ satisfies constraints (7) and (8) and simplifies the formulation to only consisting of one type of variable. Constraints (7) are now redundant and can be

removed from the formulation. The relaxed master problem becomes:

$$\max \quad \sum_{l \in L} \sum_{p \in P^l} x_p^l \tag{9}$$

$$\text{s.t.} \quad \sum_{l \in L} \sum_{p \in P^l} \delta_e^p x_p^l \leq u_e \quad \forall e \in E \tag{10}$$

$$\sum_{p \in P^l} \frac{x_p^l}{u_p} \leq k \qquad \forall l \in L \tag{11}$$

$$x_p^l \geq 0 \qquad \forall l \in L, \forall p \in P^l \tag{12}$$

3.1 Pricing problem

Let $\pi \geq \mathbf{0}$ and $\lambda \geq \mathbf{0}$ be the dual variables for constraints (10) and (11). The reduced cost for a path $p \in P^l$ for a commodity $l \in L$ is:

$$c_P^l = 1 - \sum_{e \in E} \delta_e^p \pi_e - \frac{\lambda^l}{u_p} \tag{13}$$

The pricing problems generate columns with positive reduced cost for each commodity l. Now, λ^l is a constant when l is fixed so finding a column with positive reduced cost (if any exists) is equivalent to solving the shortest path problem:

$$\sum_{e \in E} \delta_e^p \pi_e \leq 1 - \frac{\lambda^l}{u_p}, \quad \forall l \in L, \ \forall p \in P^l$$

The path capacity u_p is not known until the path has been generated. Hence, we set fixed bounds on u_p. We know that the capacity can be set to at most $|E|$ different values (one for each different $u_e : e \in E$), hence the pricing problems can be solved by considering at most $|E|$ shortest path problems. The pricing problems can now be defined as solving the shortest path problem defined on costs $\pi \geq \mathbf{0}$ for edges with $u_e \geq u_p$ for each different $u_e : e \in E$. This can be done in polynomial time by using, e.g., Dijkstra's algorithm.

3.2 Branching scheme – forbidding edge sequences

The branching scheme consists of forbidding edge sequences. Let the divergence vertex for a commodity be defined as the first vertex with one incoming path and several outgoing paths. If the number of paths emanating from the divergence vertex for some commodity l is greater than k then branching can be applied. For each emanating path p we find the first edges of p, which make p different from the remaining emanating paths. This is denoted the unique edge sequence for p. Each unique edge sequence is forbidden in a branching child. If more than $k+1$ paths emanate from the divergence vertex, then we let some branching children consist of more than one unique edge sequence such that the number of branching children is always equal to $k+1$. The reason for this is to reduce the width of the search tree. It is legal to let a branching child forbid several unique edge sequences, because all combinations of k paths from the emanating paths are available in at least one other branching child.

An illustration of the branching strategy is seen in Figure 1. A graph with four vertices is given and a commodity with source s and destination t is to be routed using at most

two paths. In the current solution three paths are used: $p_1 = \{e_1, e_4, e_5\}, p_2 = \{e_1, e_3, e_5\}$, and $p_3 = \{e_2, e_3, e_5\}$. Assume that the optimal solution consists of path p_1 and p_3. Now $k + 1$ subpaths are found: $\{e_1, e_3\}$, $\{e_1, e_4\}$ and $\{e_2\}$. The optimal solution is found in the branching child, which forbids the use of edge sequence $\{e_1, e_3\}$.

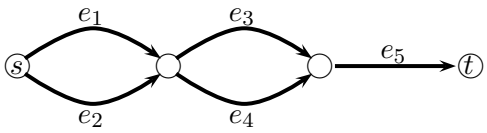


Figure 1: A graph used to illustrate the branching scheme. The graph consists of four vertices, the source vertex is denoted s, and the destination vertex t. Edges are e_1, e_2, e_3, e_4, and e_5.

The branching scheme changes the pricing problem. When solving the shortest path problem we need to ensure that we do not use the forbidden edge sequences. The shortest path problem with forbidden paths is a polynomial problem and can be solved by applying a shortest path algorithm to an extended graph, see Villeneuve and Desaulniers [10].

4 A new 2-index branch-and-price algorithm (2BP')

The 2BP' algorithm only differs from the 2BP algorithm in the branching scheme. The master problem (9)–(12) is the same and the reduced cost is given by (13).

4.1 Branching

This branching scheme resembles the branching strategy of Cook et al. [3] and is based on the idea of forbidding or forcing the use of a certain path p' for a fixed commodity $l \in L$. This corresponds to setting $y^l_{p'} = 0$ or $y^l_{p'} = 1$, respectively, in the non-relaxed master problem. In the remainder of this section a fixed commodity $l \in L$ is assumed.

The effect of the branching scheme on the non-relaxed master problem, specifically constraint (8) is considered:

$$\sum_{p \in P} y^l_p \leq k$$

In both the case that $y^l_{p'} = 0$ or $y^l_{p'} = 1$ the variable can be left out of the constraint. If $y^l_{p'} = 1$ then the constraint is rewritten as

$$\sum_{p \in P \setminus \{p'\}} y^l_p \leq k - 1$$

Now, the effect of the branching scheme on the relaxed master problem, specifically constraint (11) is considered:

$$\sum_{p \in P^l} \frac{x^l_p}{u_p} \leq k$$

When path p' is forbidden for commodity l then $x^l_{p'} = 0$. When use of path p' is forced then we set $x^l_{p'} > 0$ and constraint (11) is rewritten as

$$\sum_{p \in P^l \setminus \{p'\}} \frac{x^l_p}{u_p} \leq k - 1 \tag{14}$$

This is stronger than the original constraint when $x^l_{p'} < u_{p'}$, hence the bound of the branching child is strengthened in this case.

The number of branching children varies according to the current fractional solution. Assume that the current solution consists of $k + \alpha, \alpha > 0$ paths for commodity l. If a path in the current solution carries as much flow as possible, i.e., $x^l_p = u_p$, then forcing the use of path p has no effect because (14) is not violated.

Since the current fractional solution is a feasible solution to the relaxed master problem constraints (11) are satisfied. Hence, at least $\alpha + 1$ paths have $x^l_p < u_p$ (otherwise the sum $\sum_{p \in P} x^l_p/u_p$ would exceed k). An optimal solution may consist of paths not part of the current fractional solution. Thus, we cannot generate $\alpha+1$ branching children, where the use of exactly one path is forced in each child. Rather, $\alpha+2$ children should be generated: Each of the first $\alpha + 1$ branching children forces the use of exactly one path p with $x^l_p < u_p$, and the last branching child forbids the use of all $\alpha + 1$ paths.

The first $\alpha + 1$ children cause symmetry in the solution space; several solutions in one branching child can also be found in the other children, especially when several of the $\alpha + 1$ paths are part of the solutions. The first $\alpha + 1$ children are thus changed into forcing *and* forbidding the use of certain paths. Consider the $\alpha + 1 = 3$ branching children b_1, b_2, and b_3, forcing the use of path p_1, p_2, and p_3, respectively. Child b_1 is unaltered and forces the use of p_1. Child b_2 forces the use of p_2 and forbids the use of p_1. In this way, the solution using p_1 and p_2 is only available in the subtree of b_1. Similarly, child b_3 forces the use of p_3 and forbids the use of p_1 and p_2.

In practice we would rather add a cut than rewrite constraints (11) when the use of a path is forced. Recall inequality (14) when forcing the use of path p'. This inequality is denoted the branch cut. Let $\omega_{bl} \geq 0$ be the dual of branch cut b for commodity l. The resulting reduced cost for path $p \in P^l$ for commodity $l \in L$ is

$$c^l_p = 1 - \sum_{e \in E} \delta^e_p \pi_e - \frac{\lambda_l}{u_p} - \sum_{b \in B} \frac{\delta^b_p \omega_{bl}}{u_p} \tag{15}$$

The extra dual cost ω_{bl} is subtracted from the reduced costs for all new paths for commodity l; this is similar to how λ_l is handled. Hence, the branch cut does not affect edge weights or path properties in the graph of the pricing problem. The pricing problem must, however, be able to avoid using forbidden paths as before.

5 Computational results

A computational evaluation is performed on a dual 2.66GHz Intel® Xeon® X5355 machine with 16 GB of RAM. Note that CPU times in the following stem from using one core only.

We have tested three algorithms; the 3BP extended with a heuristic to reach feasible solutions faster, the 2BP, and the 2BP'. We implemented all three algorithms using the

Name	$\|V\|$	$\|E\|$	$\|L\|$
Random5-35	5	35	1
Random10-45	10	45	1
Random15-60	15	60	1
Random20-140	20	140	1
tg10-2	12	40	1
tg20-2	22	80	1
tg40-1	42	154	1
tg40-5	42	154	1
tg80-1	82	322	1
tg100-2	102	400	1
Random10-40	10	40	3
Random11-42	11	42	11
Random20-80	20	80	20
Random22-56	22	56	22

Table 1: Sizes of test instances. First column denotes the instance name, then follows the number of vertices, the number of edges, and finally the number of commodities.

framework of COIN [2] with `ILOG CPLEX 10.2` as LP-solver. Computations concerning the selection of branching candidates and branching children are handled by COIN.

The three solution methods are tested on benchmark instances from the literature Truffot and Duhamel [8]: The `Random` instances are randomly generated and the `tg` instances are generated by the Transit Grid generator[1] using topologies from transportation networks. See Table 1 for details.

Two different types of tests have been performed. First the three exact algorithms are computationally evaluated on the proposed instances and results are compared. Then we examine if the 3BP and either of the 2BP and 2BP' algorithms give good heuristic solutions by terminating each test run once the root node has been computed (when omitting branching the 2BP and the 2BP' algorithms are identical).

5.1 Optimal approach

The three algorithms are computationally evaluated on the proposed instances. Results for the single-commodity `Random` instances are summarized in Table 2 and results for the single-commodity `tg` instances are summarized in Table 3. The multi-commodity instances are all of the `Random` type and results are summarized in Table 4.

In the tables the first column holds the name of the problem instance, the second column holds the value of k and the third column holds the optimal value. Then follows the size and depth of the search tree, the number of generated variables, the gap in percent between the upper and lower bound, and the time in seconds spent on solving the instance for the 3BP, the 2BP, and the 2BP' algorithms, respectively. If a test run is marked with "-" then it has run out of memory. If the gap is also marked with "-" then no lower bound was found. The total number of times each algorithm has best performance, is found at the bottom of each table. Also, for each instance the best performance is written in **bold**.

The 2BP algorithm performs much better than the 3BP algorithm for the Minimum Cost MCkFP Gamst et al. [5]; however, this is generally not the case for the MCkMFP. Although

[1] `http://www.informatik.uni-trier.de/~naeher/Professur/research/generators/maxflow/tg/index.html`

			3BP					2BP					2BP'				
Problem	**k**	**z***	**size**	**depth**	**vars**	**gap**	**time**	**size**	**depth**	**vars**	**gap**	**time**	**size**	**depth**	**vars**	**gap**	**time**
Random5-35	1	66	1	0	5	0.00%	**0.00**	1	0	5	0.00%	**0.00**	1	0	5	0.00%	**0.00**
	2	128	1	0	14	0.00%	**0.00**	1	0	7	0.00%	**0.00**	1	0	7	0.00%	**0.00**
	3	182	1	0	27	0.00%	0.01	1	0	9	0.00%	**0.00**	1	0	9	0.00%	**0.00**
	4	223	13	6	48	0.00%	0.01	1	0	12	0.00%	**0.00**	1	0	12	0.00%	**0.00**
	5	262	19	9	60	0.00%	0.03	1	0	12	0.00%	**0.00**	1	0	12	0.00%	**0.00**
	6	297	21	10	78	0.00%	0.03	1	0	14	0.00%	0.01	1	0	14	0.00%	**0.00**
	7	326	67	12	98	0.00%	0.10	1	0	14	0.00%	**0.00**	1	0	14	0.00%	**0.00**
	8	326	1	0	104	0.00%	**0.00**	1	0	11	0.00%	0.01	1	0	11	0.00%	**0.00**
Random10-45	1	73	1	0	6	0.00%	**0.00**	1	0	5	0.00%	**0.00**	1	0	5	0.00%	**0.00**
	2	142	5	2	15	0.00%	**0.01**	4	1	9	0.00%	**0.01**	8	2	9	0.00%	**0.01**
	3	209	9	3	33	0.00%	**0.02**	21	3	15	0.00%	0.03	20	3	12	0.00%	**0.02**
	4	260	45	17	68	0.00%	0.08	411	12	24	0.00%	0.56	34	4	20	0.00%	**0.03**
	5	306	369	22	102	0.00%	0.80	23599	18	34	0.00%	44.96	40	4	20	0.00%	**0.07**
	6	345	973	26	137	0.00%	2.90	¿427099	¿26	39	2.36%	-	135	6	26	0.00%	**0.22**
	7	381	4281	36	219	0.00%	16.55	¿354551	¿22	46	-%	-	313	8	34	0.00%	**0.64**
	8	413	22985	43	265	0.00%	102.51	¿431299	¿29	46	2.93%	-	606	9	40	0.00%	**1.31**
	9	429	¿110199	¿58	380	6.43%	-	¿388228	¿26	60	-%	-	2507	11	46	0.00%	**5.97**
	10	451	¿104999	¿57	448	5.74%	-	¿456699	¿41	74	6.57%	-	2355	12	46	0.00%	**5.91**
Random15-60	1	86	1	0	5	0.00%	**0.00**	1	0	6	0.00%	**0.00**	1	0	6	0.00%	**0.00**
	2	163	1	0	16	0.00%	**0.00**	1	0	8	0.00%	**0.00**	1	0	8	0.00%	**0.00**
	3	221	9	3	34	0.00%	**0.02**	41	6	15	0.00%	0.06	12	2	12	0.00%	**0.02**
	4	248	111	10	70	0.00%	0.32	¿100454	¿26	50	-%	-	111	6	20	0.00%	**0.22**
	5	268	557	18	101	0.00%	551.83	¿176599	¿29	52	2.86%	-	322	7	29	0.00%	**0.76**
	6	287	419	21	135	0.00%	1.59	¿277801	¿31	45	2.74%	-	354	9	30	0.00%	**0.79**
	7	295	19097	35	194	0.00%	72.91	¿387565	¿23	49	-%	-	836	10	27	0.00%	**1.74**
	8	301	¿88799	¿47	231	2.90%	-	¿413343	¿33	55	2.90%	-	4995	11	30	0.00%	**11.32**
	9	306	¿153099	¿51	229	1.29%	-	¿547079	¿28	48	-%	-	2263	11	19	0.00%	**4.42**
Random20-140	1	81	1	0	4	0.00%	**0.00**	1	0	5	0.00%	**0.00**	1	0	5	0.00%	**0.00**
	2	158	1	0	14	0.00%	**0.00**	1	0	7	0.00%	**0.00**	1	0	7	0.00%	**0.00**
	3	228	1	0	30	0.00%	0.02	1	0	11	0.00%	**0.00**	1	0	11	0.00%	**0.00**
	4	253	9935	31	103	0.00%	75.25	¿41444	¿42	68	-%	-	90	18	67	0.00%	**1.04**
	5	274	¿39999	¿41	146	1.86%	-	¿68299	¿66	87	1.86%	-	819	22	51	0.00%	**12.65**
	6	294	¿30199	¿61	184	1.78%	-	¿60299	¿86	107	1.78%	-	¿14106	¿32	113	1.78%	-
	7	-	¿28999	¿70	227	1.81%	-	¿75894	¿46	91	-%	-	¿14299	¿32	109	1.69%	-
	8	319	¿30599	¿80	267	1.91%	-	¿94699	¿101	120	1.91%	-	4028	22	29	0.00%	**52.95**
	9	325	¿39599	¿93	315	0.84%	-	¿108990	¿63	105	-%	-	130	9	25	0.00%	**0.32**
	10	327	2907	109	326	0.00%	19.15	¿272685	¿49	68	0.61%	-	17	3	22	0.00%	**0.02**
	11	327	1325	86	301	0.00%	8.75	49	3	22	0.00%	**0.03**	20	5	20	0.00%	**0.03**
Best							**11**					**14**					**36**

Table 2: Results from solving the single-commodity `Random` instances exactly.

			3BP					2BP					2BP'				
Problem	**k**	**z***	**size**	**depth**	**vars**	**gap**	**time**	**size**	**depth**	**vars**	**gap**	**time**	**size**	**depth**	**vars**	**gap**	**time**
tg10-2	1	389	1	0	5	0.00%	**0.00**	1	0	4	0.00%	**0.00**	1	0	4	0.00%	**0.00**
	2	557	215	13	26	0.00%	0.21	355	14	10	0.00%	0.21	41	5	11	0.00%	**0.04**
	3	716	553	19	58	0.00%	0.70	39505	20	28	0.00%	32.49	53	5	15	0.00%	**0.06**
	4	815	83	17	52	0.00%	0.10	6	1	8	0.00%	**0.00**	5	1	8	0.00%	**0.00**
	5	815	1	0	40	0.00%	**0.00**	1	0	8	0.00%	**0.00**	1	0	8	0.00%	**0.00**
tg20-2	1	385	1	0	4	0.00%	**0.00**	1	0	4	0.00%	**0.00**	1	0	4	0.00%	**0.00**
	2	643	1	0	10	0.00%	**0.00**	1	0	6	0.00%	**0.00**	1	0	6	0.00%	**0.00**
	3	832	5	2	33	0.00%	0.04	1	0	10	0.00%	**0.00**	1	0	10	0.00%	**0.00**
	4	853	1	0	40	0.00%	0.01	1	0	10	0.00%	**0.00**	1	0	10	0.00%	**0.00**
tg40-1	1	517	1	0	5	0.00%	**0.00**	1	0	5	0.00%	0.01	1	0	5	0.00%	**0.00**
	2	750	5	2	16	0.00%	0.07	4	1	10	0.00%	**0.02**	10	3	12	0.00%	0.07
	3	908	¿9999	¿40	96	2.61%	-	¿83282	¿61	94	-%	-	231	11	21	0.00%	**3.32**
	4	994	¿7799	¿57	143	1.00%	-	¿82770	¿45	64	-%	-	893	18	33	0.00%	**25.15**
	5	1004	15	7	65	0.00%	0.09	703	27	20	0.00%	1.41	11	2	18	0.00%	**0.03**
	6	1004	1	0	96	0.00%	0.03	29	3	13	0.00%	**0.02**	43	6	13	0.00%	0.07
tg40-5	1	487	1	0	8	0.00%	0.01	1	0	6	0.00%	**0.00**	1	0	6	0.00%	**0.00**
	2	828	¿20599	¿46	80	4.11%	-	¿64248	¿45	57	5.70%	-	144	9	23	0.00%	**1.49**
	3	1062	¿17299	¿59	139	0.28%	-	¿77103	¿44	65	-%	-	276	8	22	0.00%	**4.20**
	4	1078	181	47	68	0.00%	**0.61**	¿148934	¿22	50	-%	-	1520	21	22	0.00%	26.53
	5	1078	3	1	90	0.00%	**0.03**	61	4	16	0.00%	0.04	76	20	16	0.00%	1.72
tg80-1	1	549	1	0	7	0.00%	0.04	1	0	6	0.00%	**0.02**	1	0	6	0.00%	**0.02**
	2	984	1591	29	80	0.00%	65.22	2308	22	25	0.00%	59.16	288	11	39	0.00%	**8.72**
	3	1411	¿2199	¿36	162	3.85%	-	¿51476	¿49	107	-%	-	1914	10	38	0.00%	**110.38**
tg100-2	1	530	1	0	7	0.00%	0.07	1	0	6	0.00%	0.03	1	0	6	0.00%	**0.02**
	2	1007	3	1	16	0.00%	0.20	1	0	8	0.00%	**0.04**	1	0	8	0.00%	**0.04**
	3	1407	¿1099	¿31	115	0.39%	-	¿29087	¿60	113	-%	-	229	6	51	0.00%	**29.14**
	4	1768	¿1499	¿72	234	1.51%	-	¿56256	¿40	167	-%	-	2118	9	82	0.00%	**284.41**
Best							**7**					**12**					**23**

Table 3: Results from solving the `tg` instances exactly.

			3BP					2BP					2BP'				
Problem	**k**	**z***	**size**	**depth**	**vars**	**gap**	**time**	**size**	**depth**	**vars**	**gap**	**time**	**size**	**depth**	**vars**	**gap**	**time**
Random10-40	1	110	5	2	18	0.00%	0.02	5	2	15	0.00%	**0.00**	4	1	14	0.00%	0.01
	2	194	1	0	26	0.00%	**0.00**	34	5	21	0.00%	0.04	4	1	18	0.00%	0.01
	3	258	183	18	80	0.00%	0.39	213	12	24	0.00%	0.18	50	6	23	0.00%	**0.06**
	4	293	695	36	129	0.00%	2.23	2956	16	41	0.00%	4.25	112	7	32	0.00%	**0.20**
	5	309	1989	30	176	0.00%	7.91	¿253716	¿25	56	1.24%	**1.06**	561	12	39	0.00%	**1.06**
	6	318	15905	36	253	0.00%	73.84	¿610006	¿24	59	1.35%	-	1294	13	60	0.00%	**2.63**
	7	321	¿153199	¿56	286	0.01%	-	¿335959	¿26	54	-%	-	26182	18	47	0.00%	**57.10**
	8	323	113	37	299	0.00%	**0.52**	2008	14	37	0.00%	1.23	2051	15	36	0.00%	2.43
	9	323	333	49	318	0.00%	1.38	11	1	32	0.00%	**0.01**	18	5	32	0.00%	0.02
Random11-42	1	291	5	2	29	0.00%	0.02	7	3	28	0.00%	**0.01**	7	2	27	0.00%	0.02
	2	343	1	0	50	0.00%	**0.01**	7	2	27	0.00%	**0.01**	6	1	27	0.00%	**0.01**
	3	344	1	0	75	0.00%	**0.00**	1	0	26	0.00%	**0.00**	1	0	26	0.00%	**0.00**
	4	344	1	0	100	0.00%	**0.00**	1	0	26	0.00%	**0.00**	1	0	26	0.00%	**0.00**
Random20-80	1	347	7	3	55	0.00%	0.06	3	1	51	0.00%	**0.02**	7	2	53	0.00%	0.04
	2	553	3	1	100	0.00%	0.03	4	1	50	0.00%	0.02	4	1	51	0.00%	**0.01**
	3	584	1063	24	188	0.00%	6.14	57	7	59	0.00%	**0.16**	1020	16	62	0.00%	3.45
	4	601	5599	33	277	0.00%	40.05	1041	10	60	0.00%	**2.02**	¿81550	¿548	601	2.01%	-
	5	617	13291	44	340	0.00%	117.96	4363	14	66	0.00%	**7.35**	49695	34	67	0.00%	198.61
	6	621	¿48999	¿40	380	0.01%	-	3998	11	63	0.00%	**6.42**	32552	29	58	0.00%	100.08
	7	626	413	37	412	0.00%	3.48	17	2	57	0.00%	**0.02**	116	14	57	0.00%	0.22
	8	626	1	0	440	0.00%	0.03	1	0	57	0.00%	**0.01**	1	0	57	0.00%	**0.01**
Random22-56	1	365	9	3	52	0.00%	**0.02**	7	3	42	0.00%	**0.02**	7	2	44	0.00%	**0.02**
	2	389	11	4	81	0.00%	0.02	10	3	42	0.00%	0.02	9	3	42	0.00%	**0.01**
	3	407	1	0	108	0.00%	**0.01**	1	0	41	0.00%	**0.01**	1	0	41	0.00%	**0.01**
	4	407	1	0	144	0.00%	0.01	1	0	41	0.00%	**0.00**	1	0	41	0.00%	**0.00**
Best							**7**					**17**					**14**

Table 4: Results from solving the multi-commodity instances exactly.

the number of times the algorithm has best performance is larger for the 2BP, the 3BP algorithm is capable of solving more instances. The change of objective function has a great impact on the problem; the algorithms always try to push as much flow through the network as possible, thus potentially exploiting the somewhat weakly formulated bound on the number of used paths. The formulation has less impact on the minimum cost problem because it may not always be beneficial to increase the number of used paths. The 2BP algorithm suffers from large search trees because of the existence of potentially many solutions using more than k paths per commodity and because the branching scheme allows much symmetry in the branching children. The 2BP algorithm, however, performs somewhat better than the 3BP for the multi-commodity `Random` instances with respect to running times.

The 2BP' algorithm generally performs much better than 3BP algorithm. Exceptions are `tg40-5`, $k = 4$ and `Random20-80`, $k = 5$, which the 2BP' algorithm spends more time on solving. Furthermore, 2BP' is unable find an optimal solution for `Random20-80`, $k = 4$. For the far majority of test instances, however, the 2BP' algorithm is capable of finding an optimal solution in little time, even when the 3BP algorithm shows great difficulty. The 2BP' algorithm generally also generates smaller gaps for instances, which are not solved to optimality. Reasons are that the search tree sizes are generally smaller for the 2BP', the number of variables in the master problem is smaller, and much symmetry is eliminated because of the lacking h-indices.

The 2BP' algorithm generally also performs much better than the 2BP algorithm. Exceptions are `Random20-80`, $k = 4, 5$, and 6 where the 2BP has overall best performance. The reason for the generally superior performance of the 2BP' algorithm is that the branching scheme gives better bounds in the branching children: forcing the use of a path is much stronger than forbidding a path. Also forbidding the use of all paths with positive flow is stronger than forbidding a subset of the paths.

All three algorithms suffer from the same weakness in the formulation, specifically the bounding of the number of used paths per commodity: constraints (3) for the 3BP and (11) for the 2BP and the 2BP' algorithms. Because the objective is to maximize the total amount of flow, the algorithms are very likely to exceed k paths per commodity whenever the mentioned constraints are not tight. The constraints will rarely be tight, especially when several paths share the same edges and the corresponding x_p^l/u_p then can become much smaller than one. The 2BP' reduces this problem to some extend with the branching cut (14).

5.2 Heuristic approach

The three exact algorithms presented can be used as heuristics by only computing the root node and then returning the best feasible solution. The approach of only computing the root node does not guarantee a polynomial running time, since an exponential number of columns potentially needs to be added in the root. In practice, however, we expect low running times.

The heuristic usage of the 3BP algorithm is denoted 3HEUR. Because no branching occurs the heuristic usage of the 2BP and the 2BP' algorithms is identical and is denoted 2HEUR. Truffot and Duhamel [8] argue that the 3-index and 2-index formulations are equivalent, also after LP-relaxation and elimination of the binary variables. Even though the formulations give the same bounds, we may not reach the same feasible solutions in the root node. Hence we investigate the performance of 3HEUR and 2HEUR empirically.

The 2HEUR may give infeasible solutions where more than k paths are used for each

commodity. In this case we try to move the flow between the paths in order to find a feasible solution using at most k paths for each commodity. For each commodity the approach investigates all paths in the current fractional solution and greedily assigns flow to the path having the highest capacity. The steps of the approach are:

1: **for** (each commodity) **do**
2: Sort all the paths in the current fractional solution according to decreasing capacity
3: **for** (each path in the sorted list, until flow is assigned to k paths) **do**
4: Assign as much flow as possible to the path
5: Subtract the assigned flow from the capacity of each edge on the path
6: **end for**
7: **end for**

Including this flow-moving approach in 2HEUR gives the final heuristic denoted 2HEUR'. It is noted that including the flow-moving approach in the exact 2BP and 2BP' approaches does not improve performance; see the tables at `http://www.diku.dk/~gamst/heuristic_results.pdf` for documentation.

All three heuristics 3HEUR, 2HEUR, and 2HEUR' are evaluated on the previously proposed instances. Test results are summarized in tables 5, 6, and 7.

The first column of each table holds the name of the problem instance, the second column holds the value of k, and the third column holds the optimal value. Then, follows for each of the algorithms 3HEUR, 2HEUR, and 2HEUR'; the number of iterations, the gap between the heuristic and the optimal value, and the time in seconds spent on solving the instance. An entry marked with "-" indicates that no feasible solution was found. The average number of iterations, gap, and time usage are given at the bottom of each table.

The results show that the 3HEUR algorithm often gives poor heuristic solutions with gaps of up to 94%. For three multi-commodity `Random` instances the 3BP algorithm is even unable to find a feasible solution in the root node. The 2HEUR algorithm generally finds much better solution values than the 3HEUR algorithm. The 2HEUR', however, shows superior performance by solving the majority of the instances to optimality and with the largest gap of those not solved being 20%. All heuristics have very low running times and terminate in less than a second.

			3HEUR			2HEUR			2HEUR'		
Problem	**k**	**z***	**iter.**	**gap**	**time**	**iter.**	**gap**	**time**	**iter.**	**gap**	**time**
Random5-35	1	66	5	0.00	0.00	3	0.00	0.00	3	0.00	0.01
	2	128	7	0.00	0.00	5	0.00	0.00	5	0.00	0.00
	3	182	8	0.00	0.00	7	0.00	0.00	7	0.00	0.00
	4	223	12	16.60	0.00	10	0.00	0.00	10	0.00	0.00
	5	262	12	54.96	0.00	10	0.00	0.00	10	0.00	0.00
	6	297	13	80.37	0.00	12	0.00	0.00	12	0.00	0.00
	7	326	14	54.29	0.00	12	0.00	0.00	12	0.00	0.00
	8	326	13	0.00	0.01	11	0.00	0.00	11	0.00	0.00
Random10-45	1	73	6	0.00	0.00	4	0.00	0.00	4	0.00	0.00
	2	142	7	12.68	0.00	6	12.68	0.00	6	0.00	0.00
	3	209	10	22.00	0.01	10	15.31	0.00	10	0.00	0.00
	4	260	13	65.38	0.01	13	18.08	0.00	13	0.00	0.00
	5	306	15	75.82	0.01	17	46.73	0.00	17	0.00	0.00
	6	345	17	73.91	0.02	19	43.48	0.00	19	0.00	0.00
	7	381	23	76.38	0.02	21	47.77	0.00	21	8.40	0.01
	8	413	24	78.21	0.02	23	48.18	0.00	23	6.78	0.01
	9	429	30	79.02	0.04	30	37.06	0.00	30	1.40	0.00
	10	451	35	80.04	0.05	38	36.59	0.01	38	0.00	0.01
Random15-60	1	86	5	0.00	0.00	6	0.00	0.00	6	0.00	0.00
	2	163	8	0.00	0.01	8	0.00	0.00	8	0.00	0.00
	3	221	10	55.24	0.01	11	85.52	0.00	11	16.74	0.00
	4	248	13	87.50	0.01	18	56.04	0.01	18	10.01	0.00
	5	268	16	57.49	0.02	20	51.49	0.00	20	5.97	0.00
	6	287	18	93.38	0.02	22	50.52	0.01	22	6.62	0.00
	7	295	20	85.05	0.02	23	46.44	0.01	23	13.90	0.00
	8	301	19	59.47	0.01	21	46.84	0.00	21	17.94	0.00
	9	306	18	60.13	0.01	18	39.87	0.00	18	19.93	0.00
Random20-140	1	81	4	0.00	0.00	4	0.00	0.00	4	0.00	0.00
	2	158	7	0.00	0.02	6	0.00	0.00	6	0.00	0.00
	3	228	10	0.00	0.02	10	0.00	0.00	10	0.00	0.00
	4	253	12	82.48	0.03	13	0.00	0.00	13	0.00	0.01
	5	274	16	84.69	0.04	16	1.46	0.01	16	0.00	0.01
	6	294	18	84.69	0.04	24	69.05	0.01	24	3.40	0.01
	9	325	22	86.15	0.04	24	44.92	0.01	24	0.31	0.01
	10	327	21	86.24	0.01	19	3.67	0.00	19	3.67	0.00
	11	327	20	86.24	0.01	19	0.61	0.00	19	0.61	0.00
Sum			14	49.40	0.01	15	22.29	¡0.01	15	3.21	¡0.01

Table 5: Results from solving the single-commodity `Random` instances heuristically, where each algorithm terminates after having evaluated the root node only.

			3HEUR			2HEUR			2HEUR'		
Problem	**k**	**z***	**iter.**	**gap**	**time**	**iter.**	**gap**	**time**	**iter.**	**gap**	**time**
tg10-2	1	389	5	0.00	0.00	4	0.00	0.00	4	0.00	0.00
	2	557	6	0.00	0.00	6	0.00	0.00	6	0.00	0.00
	3	716	9	0.00	0.00	10	15.22	0.00	10	0.00	0.00
	4	815	8	0.00	0.00	8	15.83	0.01	8	0.00	0.00
	5	815	8	0.00	0.00	8	0.00	0.00	8	0.00	0.00
tg20-2	1	385	4	0.00	0.00	4	0.00	0.00	4	0.00	0.00
	2	643	5	0.00	0.00	6	0.00	0.00	6	0.00	0.00
	3	832	11	18.03	0.00	10	0.01	0.00	10	0.00	0.01
	4	853	10	0.00	0.01	10	0.00	0.00	10	0.00	0.00
tg40-1	1	517	5	0.00	0.01	5	0.00	0.01	5	0.00	0.01
	2	750	7	72.13	0.01	9	61.33	0.01	9	0.00	0.01
tg40-5	1	487	8	0.00	0.00	6	0.00	0.00	6	0.00	0.01
tg80-1	1	549	7	0.00	0.04	6	0.00	0.02	6	0.00	0.02
	2	984	11	52.74	0.14	14	14.63	0.06	14	0.00	0.06
tg100-2	1	530	7	0.00	0.07	6	0.00	0.02	6	0.00	0.02
	2	1007	8	28.10	0.15	8	0.00	0.04	8	0.00	0.03
Sum			7	10.69	0.03	8	6.69	0.01	8	0.00	0.01

Table 6: Results from solving the `tg` instances heuristically, where each algorithm terminates after having evaluated the root node only.

			3HEUR			2HEUR			2HEUR'		
Problem	**k**	**z***	**iter.**	**gap**	**time**	**iter.**	**gap**	**time**	**iter.**	**gap**	**time**
Random10-40	1	110	6	31.82	0.00	5	20.00	0.01	5	17.27	0.00
	2	194	6	0.00	0.00	9	60.82	0.00	9	0.00	0.00
	3	258	11	80.62	0.01	11	69.38	0.00	11	6.59	0.00
	4	293	14	66.21	0.02	15	36.52	0.01	15	7.17	0.01
	5	309	16	67.96	0.02	19	34.95	0.00	19	8.41	0.01
	6	318	21	68.89	0.03	25	33.02	0.00	25	5.97	0.01
	7	321	17	84.42	0.02	21	24.61	0.00	21	1.56	0.01
	8	323	21	84.52	0.01	20	22.29	0.00	20	4.34	0.00
	9	323	20	84.52	0.01	21	9.29	0.00	21	0.00	0.00
Random11-42	1	291	6	6.19	0.01	6	6.19	0.00	6	0.00	0.01
	2	343	4	0.00	0.00	5	19.53	0.00	5	4.08	0.00
	3	344	4	0.00	0.00	5	0.00	0.01	5	0.00	0.00
	4	344	4	0.00	0.00	5	0.00	0.00	5	0.00	0.00
Random20-80	1	347	6	25.36	0.01	6	16.14	0.01	6	0.00	0.01
	2	553	7	35.80	0.02	7	15.91	0.01	7	0.00	0.01
	3	584	9	-	0.02	9	7.53	0.00	9	0.00	0.01
	4	601	12	-	0.03	12	7.65	0.01	12	0.00	0.01
	5	617	14	-	0.04	16	4.05	0.02	16	2.27	0.00
	6	621	12	58.29	0.03	14	0.64	0.01	14	0.00	0.01
	7	626	12	58.63	0.03	14	0.96	0.01	14	0.80	0.01
	8	626	12	0.00	0.03	14	0.00	0.01	14	0.00	0.01
Random22-56	1	365	6	0.00	0.01	5	0.00	0.00	5	0.00	0.00
	2	389	6	1.54	0.00	5	1.54	0.00	5	0.00	0.00
	3	407	6	0.00	0.01	5	0.00	0.00	5	0.00	0.01
	4	407	6	0.00	0.00	5	0.00	0.01	5	0.00	0.01
Sum*			9	34.31	0.01	11	15.64	¡0.01	11	2.34	¡0.01

Table 7: Results from solving the multi-commodity `Random` instances heuristically, where each algorithm terminates after having evaluated the root node only. *) sum is only over the instances where all heuristics found a feasible solution.

6 Conclusion

Two exact solution methods for the MCkMFP problem have been introduced. They are both based on Dantzig-Wolfe decomposition, where the master problem is a 2-index formulation merging paths for commodities into an overall solution. The two methods differ in their branching schemes: the first method forbids subpaths (2BP), while the second forces or forbids the use of certain paths (2BP'). The latter also adds branching cuts to the master problem.

The 2BP and 2BP' algorithms have been implemented and compared with a leading exact algorithm from the literature denoted 3BP. Results showed that the 2BP' algorithm performs significantly better than the 2BP and the 3BP algorithms both with respect to the number of solved instances and with respect to the time usage. The main reason is that using the 2BP' algorithm gives smaller search trees, reduces the number of variables in the master problem, and eliminates some of the symmetry in the solution space.

Terminating the computations after having evaluated the root node transforms the 3BP and the 2BP/2BP' algorithms into heuristics denoted 3HEUR and 2HEUR, respectively. Because no branching occurs in this heuristic use, the 2BP and the 2BP' algorithms become identical. Test results for this approach showed that the 3HEUR does not perform well, with the majority of the solution values having gaps of up to 94%. The 2HEUR algorithm, however, showed very promising performance when including a flow-moving approach, which transforms some fractional solutions into feasible solutions. In most cases optimal solutions were found and the average solution gaps never exceeded 4%. Both heuristics terminate in less than a second for all tested instances.

All algorithms suffer from weak formulations for bounding the number of used paths per commodity. We believe that future work should concentrate on tightening these constraints.

This could be done by somehow reformulating the problem or by adding cuts. We believe that the focus should be on cuts violated in the edge-based model or the original master problem. Future work could also concentrate on finding better branching strategies for the 2-index formulation in order to further reduce the size of the search tree.

Acknowledgement
We would like to thank GlobalConnect A/S for their support of this work.

References

[1] G. Baier, E. Köhler, and M. Skutella. On the k-splittable flow problem. In *10th Annual European Symposium on Algorithms*, pages 101–113, 2002.

[2] COIN. `http://www.coin-or.org/Doxygen/Bcp/hierarchy.html`, 2007. URL `http://www.coin-or.org/Doxygen/Bcp/hierarchy.html`.

[3] W. J. Cook, W. H. Cunningham, W. R. Pulleyblank, and A. Schrijver. *Combinatorial optimization*. John Wiley & Sons, Inc, 1998.

[4] G. B. Dantzig and P. Wolfe. Decomposition principle for linear programs. *Operations Research*, 8:101–111, 1960.

[5] M. Gamst, P. N. Jensen, D. Pisinger, and C. E. M. Plum. Two- and three-index formulations of the minimum cost multicommodity. *European Journal of Operations Research (EJOR)*, 202(1):82–89, 2010.

[6] R. Koch, M. Skutella, and I. Spenke. Approximation and complexity of k-splittable flows. In *Approximation and Online Algorithms, Third International Workshop, WAOA*, pages 244–257, 2005.

[7] R. Koch, M. Skutella, and I. Spenke. Maximum k-splittable s, t -flows. *Theory of Computing Systems*, 43(1):1432–4350, 2008.

[8] J. Truffot and C. Duhamel. A branch and price algorithm for the k-splittable maximum flow problem. *Discrete Optimization*, 5(3):629–646, 2008.

[9] J. Truffot, C. Duhamel, and P. Mahey. Using branch-and-price to solve multicommodity k-splittable flow problems. In *Proceedings of International Network Optimization Conference (INOC)*, 2005.

[10] D. Villeneuve and G. Desaulniers. The shortest path problem with forbidden paths. *European Journal of Operational Research*, 165(1):97–107, 2005.

Chapter 12

Partial Path Column Generation for the Elementary Shortest Path Problem with Resource Constraints

Mads Jepsen
DTU Management Engineering, Technical University of Denmark

Bjørn Petersen
DTU Management Engineering, Technical University of Denmark

Abstract

This paper introduces a decomposition of the Elementary Shortest Path Problem with Resource Constraints (ESPPRC), where the path is composed of smaller sub paths. We show computationals result by comparing different approaches for the decomposition and compare the best of these with existing algorithms. We show that the algorithm for many instances outperforms a bidirectional labeling algorithm.

Keywords: Elementary Shortest Path With Resource Constraints, Column Generation, Dantzig-Wolfe, Vehicle Routing Problem

1 Introduction

A formal definition of the ESPPRC is as follows: Given a directed $G(V, A)$ with node set $V = \{1, .., |V|\}$, arc set $A = V \times V$, a set of resources R each with a global upper bound $W^r : r \in R$. Let c_{ij} be the cost for arc $(i, j) \in A$ and w^r_{ij} be the consumption of resource $r \in R$ when traversing arc $(i, j) \in A$. A path p is feasible if the arcs traversed on the path $A(p)$ satisfies $\sum_{(i,j)\in A(p)} w^r_{ij} \leq W^r$ for all $r \in R$. The objective is to find a feasible path p with minimum cost $\sum_{(i,j)\in A(P)} c_{ij}$ from a origin node $o \in V$ to a destination node $d \in V$.

When negative cycles are allowed in G the ESPPRC can be shown to be $\mathcal{NP}$-hard by reduction from the longest path problem, Dror [5]. Beasley and Christofides [2] gave a mathematical formulation of the problem where each node is considered a resource. Feillet et al.

Extended abstract, INOC 2009.

[6] introduced a labeling algorithm. Righini and Salani [9] proposed a bi-directional labeling and a Branch-and-Bound algorithm. Baldacci et al. [1] computed lower bounds on paths costs and used these to speed up a bi-directional labeling algorithm.

The main application of the ESPPRC is as a pricing problem when solving the Vehicle Routing Problem through Branch-Cut-and-Price. Chabrier [3] and Jepsen et al. [8] have shown this successfully for the Vehicle Routing Problem with Time Windows (VRPTW) and Baldacci et al. [1] recently for the Capacitated Vehicle Routing Problem (CVRP).

Labeling algorithms has so far been used very successfully for ESPPRC problems especially when time windows are present. However, for problem instances where the time windows are very large the state space becomes huge and labeling algorithms are no longer desirable.

Motivated by the bi-directional labeling algorithm by Righini and Salani [9] and the fact that Branch-and-Cut has been used quite successfully to solve the ESPPRC when time window like restrictions are not included (see Jepsen et al. [7]), we propose a Danzig-Wolfe decomposition approach based on a model where small sub paths called partial paths are concatenated to form the solution. Since each of the sub paths are elementary the SR-inequalities for VRPTW introduced by Jepsen et al. [8] can be used to improve the lower bound. Furthermore, valid inequalities for the ESPPRC can be used.

2 Bounded partial paths

The idea behind the following mathematical model and decomposition is that any feasible path p can be seen as a sequence of $K = \{1, \ldots, |K|\}$ partial paths $p_{ov_1}, p_{v_1v_2}, \ldots, p_{v_{k-1}d}$, where p_{ij} is a partial path from node i to node j. Each of the $|K|$ partial paths can be seen as a path through the original graph G. This leads to an alternative formulation of the ESPPRC where G is replicated $|K|$ times and arcs are added between the adjacent layers.

Let L^r be the upper bound of resource $r \in R$ on each partial path and let

$$w^r_{\max} = \max_{(i,j)\in A} w^r_{ij}$$

be the maximal resource consumption of r on a single arc. For a fixed number of partial paths $|K|$ the following relation ensures that all solutions can be obtained:

$$L^r \geq \left\lceil \frac{W^r}{|K|} \right\rceil + w^r_{\max} - 1$$

Let $\delta^+(S) = \{(i,j) \in A : i \in S\}$ denote the set of outgoing arcs of node set S and let $\delta^-(S) = \{(i,j) \in A : j \in S\}$ denote the set of ingoing arcs of S. For notational purposes let $\delta(i)$ be short for $\delta(\{i\})$ for $i \in V$. The binary variable x_{ijk} indicates if arc $(i,j) \in A$ is used in the k'th layer. The binary variable s_{ik} indicates if a partial path starts in node $i \in V$ in layer $k \in K$ and the binary variable t_{ik} indicates if a partial path ends in node $i \in V$ in layer

k. The mathematical model for ESPPRC can now be formulated as:

$$\min \sum_{k\in K}\sum_{(i,j)\in A} c_{ij}x_{ijk} \tag{1}$$

$$\texttt{s.t.} \sum_{(o,j)\in\delta^+(o)} x_{oj1} = 1 \tag{2}$$

$$\sum_{(i,d)\in\delta^-(d)} x_{id|K|} = 1 \tag{3}$$

$$\sum_{k\in K}\sum_{(i,j)\in A} x_{ijk} \leq 1 \qquad v \in V \setminus \{o,d\} \tag{4}$$

$$\sum_{k\in K}\sum_{(i,j)\in A} w^r_{ij}x_{ijk} \leq W^r \qquad r \in R \tag{5}$$

$$\sum_{k\in K}\sum_{(i,j)\in\delta^+(S)} x_{ijk} \geq \sum_{k\in K}\sum_{(i,j)\in\delta^+(s)} x_{ijk} \qquad S \subseteq V, s \in S \tag{6}$$

$$\sum_{i\in V} s_{ik} = 1 \qquad k \in K \tag{7}$$

$$t_{i,(k-1 \mod |K|)} = s_{ik} \qquad i \in V, k \in K \tag{8}$$

$$s_{ik} + \sum_{(j,i)\in\delta^-(i)} x_{jik} = t_{ik} + \sum_{(i,j)\in\delta^+(i)} x_{ijk} \qquad i \in V, k \in K \tag{9}$$

$$\sum_{(i,j)\in\delta^+(S)} x_{ijk} \geq \sum_{(i,j)\in\delta^+(s)} x_{ijk} \qquad k \in K, S \subseteq V, s \in S \tag{10}$$

$$\sum_{(i,j)\in A} x_{ijk} \leq L^{bound} \qquad k \in K \tag{11}$$

$$x_{ijk} \in \{0,1\} \qquad (i,j) \in A, k \in K \tag{12}$$

$$t_{ik}, s_{ik} \in \{0,1\} \qquad i \in V, k \in K \tag{13}$$

The objective (1) is to minimize the total cost of the path. Constraints (2), (3), and (4) ensure that no node is visited more than once and that the path starts at o and ends at d. Constraints (5) are the resource bounds and constraints (6) are the generalized subtour constraints (GSEC) which prevent cycles in a solution. Constraints (7) to (11) ensure that the partial paths are elementary, connected, and do not violate the reduced resource. $bound \in R$ is the resource chosen as the bounding resource.

In the following we will make a Danzig-Wolfe reformulation of the mathematical model, where constraints (9) to (11) form K identical sub problems. Each subproblem consists of finding a shortest path p between two arbitrary nodes in the graph. Let $\alpha^p_{ij} = 1$ iff path p uses arc (i,j), let β^p_i indicate if p starts in node i, let γ^p_i indicate iff p ends in node i, let λ^p indicate if partial path p is used, and let c_p be the cost of using path p. The master problem then becomes:

$$\min \sum_{p \in P} c_p \lambda_p \tag{14}$$

$$\texttt{s.t.} \sum_{p \in P} \sum_{(o,j) \in \delta^+(o)} \alpha_{oj}^p \lambda_p = 1 \tag{15}$$

$$\sum_{p \in P} \sum_{(i,d) \in \delta^-(d)} \alpha_{id}^p \lambda_p = 1 \tag{16}$$

$$\sum_{p \in P} \sum_{(i,j) \in A} \alpha_{ij}^p \lambda_p \leq 1 \qquad v \in V \setminus \{o, d\} \tag{17}$$

$$\sum_{p \in P} \sum_{(i,j) \in \delta^+(S)} \alpha_{ij}^p \lambda_p \geq \sum_{p \in P} \sum_{(i,j) \in \delta^+(s)} \alpha_{ij}^p \lambda_p \qquad S \subseteq V, s \in S \tag{18}$$

$$\sum_{p \in P} \sum_{(i,j) \in A} w_{ij}^r \alpha_{ij}^p \lambda_p \leq W^r \qquad r \in R \tag{19}$$

$$\sum_{p \in P} \lambda_p = |K| \tag{20}$$

$$\sum_{p \in P} \gamma_i^p \lambda_p = \sum_{p \in P} \beta_i^p \lambda_p \qquad i \in V \tag{21}$$

$$s_{ik} \in \{0, 1\} \qquad i \in V, k \in K \tag{22}$$

$$\lambda_p \in \{0, 1\} \qquad p \in P \tag{23}$$

With the exception of constraint (20) the constraints follow directly from a standard Dantzig-Wolfe reformulation. Constraint (20) substitutes the $|K|$ constraints (7) and states that we must choose $|K|$ columns corresponding to one from each layer. The master model may be too large to solve, therefore delayed column generation is used.

Let π be the duals of constraints (15), (16), and (17), let ν be the duals of constrainta (18), let σ be the duals of constraints (19), and let ρ be the duals of (21). Using standard Linear Programming theory the arc cost is set to:

$$\hat{c}_{ij} = c_{ij} - \pi_i - \sum_{r \in R} w_{ij}^r \sigma_r - \sum_{s \in S, S \subseteq V : (i,j) \in \delta^+(S)} \nu_s + \sum_{s \in S, S \subseteq V : (i,j) \in \delta^+(s)} \nu_s.$$

Let x_{ij} be a binary variable that indicates if arc $(i, j) \in A$ is used, the binary variable s_i indicates if the path starts in node $i \in V$ and the binary variable t_i indicates if the path ends in node $i \in V$. The mathematical model for the pricing problem then becomes:

$$\min \sum_{(i,j)\in A} \hat{c}_{ij} x_{ij} + \sum_{i\in V} \rho_i s_i - \sum_{i\in V} \rho_i t_i \tag{24}$$

$$\sum_{i\in V} s_i = 1 \tag{25}$$

$$\sum_{i\in V} t_i = 1 \tag{26}$$

$$s_i + \sum_{(j,i)\in\delta^-(i)} x_{ji} = t_i + \sum_{(i,j)\in\delta^+(i)} x_{ij} \qquad i \in V \tag{27}$$

$$\sum_{(i,j)\in\delta^+(S)} x_{ij} \geq \sum_{(i,j)\in\delta^+(s)} x_{ij} \qquad S \subseteq V, s \in S \tag{28}$$

$$\sum_{(i,j)\in A} x_{ij} \leq L^{bound} \tag{29}$$

$$x_{ij} \in \{0,1\} \qquad (i,j) \in A \tag{30}$$

$$s_i, t_i \in \{0,1\} \qquad i \in V \tag{31}$$

A column has negative reduced cost if it is less than the dual variable of constraint (20).

To solve the pricing problem we reformulate it as an ESPPRC. This is done by substituting the variables s_i and t_i for $i \in V$ with arcs from a super source node $\bar{s}$ and arcs to a super target $\bar{t}$ node. The new arcs are defined by arc set $\bar{A} = \{(\bar{s}, v) : v \in V\} \cup \{(v, \bar{t}) : v \in V\}$. The pricing problem then becomes solving an ESPPRC with a single resource in the graph $\bar{G}(V \cup \bar{s} \cup \bar{t}, A \cup \bar{A})$ where the cost of the new arcs are given by

$$\bar{c}_{ij} = \begin{cases} \hat{c}_{ij} & \forall (i,j) \in A \\ \rho_j & (\bar{s}, j) \in \bar{A} \\ -\rho_i & (i, \bar{t}) \in \bar{A} \end{cases}$$

The lower bound can be improved using valid inequalities for the ESPPRC polytope and valid inequalities for the master model such as the SR-inequalities by Jepsen et al. [8].

3 Implementation

The bidirectional labeling algorithm of Righini and Salani [9] have been implemented for solving the pricing problem. The Branch-Cut-And-Price algorithm is implemented in the BCP framwork from COIN [4]. CLP is used as LP solver and the GSECs are separate by solving a minimum cut problem, see Wolsey [10] for details. The SR-inequalities are separated using the algorithm proposed by Jepsen et al. [8], either the first or the last node on a partial path is not considered part of the SR-cut. Branching is done on a single arc or all arcs out of a node and is added as a cut in the master model. The constraints in the original space are:

$$\sum_{k\in K} x_{ijk} = 0 \quad \vee \quad \sum_{k\in K} x_{ijk} = 1 \qquad (i,j) \in A \tag{32}$$

$$\sum_{k\in K} \sum_{(i,j)\in\delta^+(i)} x_{ijk} = 0 \quad \vee \quad \sum_{k\in K} \sum_{(i,j)\in\delta^+(i)} x_{ijk} = 1 \qquad i \in V \tag{33}$$

The decomposed version of the branches are:

$$\sum_{p\in P} \alpha_{ij}^p \lambda_p = 0 \quad \vee \quad \sum_{p\in P} \alpha_{ij}^p \lambda_p = 1 \qquad (i,j) \in A \tag{34}$$

$$\sum_{p\in P} \sum_{(i,j)\in\delta^+(i)} \alpha_{ij}^p \lambda_p = 0 \quad \vee \quad \sum_{p\in P} \sum_{(i,j)\in\delta^+(i)} \alpha_{ij}^p \lambda_p = 1 \qquad i \in V \tag{35}$$

4 Computational Results

Based on a column generation algorithm for CVRP, some instances for the ESPPRC were generated based on CVRP instances from `www.branchandcut.org`. Tests have been performed on instances with 20-30 nodes to find a good value of L (the upper bound of the bounding resource). Based on these results setting for larger instances have been chosen.

For the instances generated the partial path have been bounded using both length and capacity. When using length it was chosen to restrict the upper limit to 6. For the capacity the path was split in pieces of at most a tenth of the total capacity. SR-inequalities were included.

We have chosen to show the result for a single instance which was representative for the instances we benchmarked on. Furthermore, the instance has the characteristics we are targeting to solve. The instance has 30 nodes, the maximal feasible path length is 23, and the capacity limit is 4500.

In Table 1 different settings for capacity and length have been compared. L is the maximal value of the bounding resource on the partial path. RB is the root bound and T is the time without SR-inequalities. RB_{SR} and T_{SR} is the root bound and time when SR-inequalities are included. All times are in seconds.

Instance	Bounded on	L	RB	T	RB_{SR}	T_{SR}
E-n30-k3-20	Capacity	2125	-192350	311.895	-192350	386.072
E-n30-k3-20	Capacity	1700	-192350	228.958	-192350	203.585
E-n30-k3-20	Capacity	1300	-192320	49.579	-192320	143.685
E-n30-k3-20	Capacity	1193	-192350	31.810	-192350	128.412
E-n30-k3-20	Capacity	1113	-192350	23.653	-192350	120.776
E-n30-k3-20	Capacity	1000	-192350	39.098	-192350	171.571
E-n30-k3-20	Capacity	900	-192350	20.173	-192350	118.271
E-n30-k3-20	Length	3	-192350	20.361	-192350	19.893
E-n30-k3-20	Length	4	-192350	44.407	-192350	50.455
E-n30-k3-20	Length	5	-192350	134.080	-192350	99.106
E-n30-k3-20	Length	6	-192350	255.236	-192350	269.989

Table 1: Comparing different schemes

From the result in Table 1 it is clear that the longer the path the poorer the algorithm performs. The main reason for this is that no matter how long the path becomes there is simply no gain in the quality of the relaxation. The value of the root bound is almost the same as the one for Branch-and-Cut, which is -192352.787. When including the SR-inequalities only a few of the settings result in a improvement of the running time.

In Table 2 we have shown the solution times for the partial path algorithm using length and capacity. T_{len} is the running time when length was used and T_{cap} is the running time

when capacity was used. The values are compared to the time of the bi-directional labeling algorithm (T_{label}) and the Branch-and-Cut algorithm (T_{BAC}) by Jepsen et al. [7]. Again, all times are in seconds.

Instance	T_{BAC}	T_{label}	T_{len}	T_{cap}
E-n30-k3-20	0.44	> 1800	19.893	20.173
B-n31-k5-17	2.07	0.22	124.492	24.178
A-n32-k5-120	0.51	0.28	32.714	7.892
A-n33-k5-31	0.45	0.01	121.440	14.477
B-n34-k5-17	2.21	72.79	290.022	32.554
B-n45-k6-54	4.63	90.3	286.978	109.011
P-n45-k5-150	0.58	0.71	19.753	15.457
P-n50-k8-19	0.94	> 1800	188.008	25.350
E-n51-k5-29	2.46	> 1800	277.645	287.746

Table 2: Characteristics of the benchmark instances

The main conclusion when comparing the results in Table 2 is that the Branch-and-Cut algorithm outperforms the other algorithms. The second observation is that the partial path algorithm is able to solve all instances within 30 minutes, labeling is only able to solve 6 out of 9. It is also worth noting that it is considerable better to bound using capacity in almost all cases compared to bounding using length. Finally, we conclude that the partial path algorithms can not compete with the Branch-and-Cut algorithm.

5 Conclusion and future research

In this paper we have introduced an alternative formulation of ESPPRC and shown how it can be solved using the Danzig-Wolfe decomposition principle. We have shown that an early prototype is better than a standard labeling algorithm, but we have not been able to show that the obtained bound is better than a Branch-and-Cut algorithm. Therefore, an open problems which has arisen during this research is if there exists an instance where the bound obtained by the partial path algorithm results in a better bound than that of an Branch-and-Cut algorithm.

References

[1] R. Baldacci, N. Christofides, and A. Mingozzi. An exact algorithm for the vehicle routing problem based on the set partitioning formulation with additional cuts. *Mathematical Programming*, 115(2):351–385, 2008. doi: 10.1007/s10107-007-0178-5.

[2] J.E. Beasley and N. Christofides. An algorithm for the resource constrained shortest path problem. *Networks*, 19:379–394, 1989. doi: 10.1002/net.3230190402.

[3] A. Chabrier. Vehicle routing problem with elementary shortest path based column generation. *Computers & Operations Research*, 33(10):2972–2990, 2006. doi: 10.1016/j.cor.2005.02.029.

[4] COIN. COIN — COmputational INfrastructure for Operations Research, 2005. http://www.coin-or.org.

[5] M. Dror. Note on the complexity of the shortest path models for column generation in VRPTW. *Operations Research*, 42(5):977–979, 1994. doi: 10.1287/opre.42.5.977.

[6] D. Feillet, P. Dejax, M. Gendreau, and C. Gueguen. An exact algorithm for the elementary shortest path problem with resource constraints: Application to some vehicle routing problems. *Networks*, 44(3):216–229, 2004. doi: 10.1002/net.v44:3.

[7] M. Jepsen, B. Petersen, and S. Spoorendonk. A branch-and-cut algorithm for the elementary shortest path problem with a capacity constraint. Technical Report 08/01, DIKU Department of Computer Science, University of Copenhagen, Denmark, 2008.

[8] M. Jepsen, B. Petersen, S. Spoorendonk, and D. Pisinger. Subset-row inequalities applied to the vehicle-routing problem with time windows. *Operations Research*, 56(2):497–511, 2008. doi: 10.1287/opre.1070.0449.

[9] G. Righini and M. Salani. Symmetry helps: bounded bi-directional dynamic programming for the elementary shortest path problem with resource constraints. *Discrete Optimization*, 3(3):255–273, 2006. doi: 10.1016/j.disopt.2006.05.007.

[10] L. A. Wolsey. *Integer Programming*. John Wiley & Sons, Inc., 1998.

Chapter 13

Partial Path Column Generation for the Vehicle Routing Problem with Time Windows

Mads Jepsen
DTU Management Engineering, Technical University of Denmark

Bjørn Petersen
DTU Management Engineering, Technical University of Denmark

Abstract

This paper presents a column generation algorithm for the Vehicle Routing Problem with Time Windows (VRPTW). Traditionally, column generation models of the VRPTW have consisted of a Set Partitioning master problem with each column representing a route, i.e., a resource feasible path starting and ending at the depot. Elementary routes (no customer visited more than once) have shown superior results on difficult instances (less restrictive capacity and time windows). However, the pricing problems do not scale well when the number of feasible routes increases, i.e., when a route may contain a large number of customers. We suggest to relax that 'each column is a route' into 'each column is a part of the giant tour'; a so-called partial path, i.e., not necessarily starting and ending in the depot. This way, the length of the partial path can be bounded and a better control of the size of the solution space for the pricing problem can be obtained.

Keywords: Vehicle Routing Problem, Column Genexration, Elementary Shortest Path Problem with Resource Constraints

1 The Vehicle Routing Problem with Time Windows

The VRPTW can formally be stated as: Given a graph $G(V, A)$ with nodes V and arcs A, a set R of resources ($R = \{\text{load}, \text{time}\}$) where each resource $r \in R$ has a lower bound a_i^r and an upper bound b_i^r for all $i \in V$ and a positive consumption τ_{ij}^r when using arc $(i, j) \in A$, find a

Extended abstract, INOC 2009.

set of routes starting and ending at the depot node $0 \in V$ satisfying all resource limits, such that the cost is minimized and all customers $C = V \setminus \{0\}$ are visited.

2-index formulation of the VRPTW In the following let c_{ij} be the cost of arc $(i,j) \in A$, x_{ij} be the binary variable indicating the use of arc $(i,j) \in A$, and T^r_{ij} be the consumption of resource $r \in R$ at the beginning of arc $(i,j) \in A$. Let $\delta^+(i)$ and $\delta^-(i)$ be the set of outgoing respectively ingoing arcs of node $i \in V$. The mathematical model of VRPTW adapted from Bard et al. [2] and Ascheuer et al. [1] is

$$\min \sum_{(i,j)\in A} c_{ij}x_{ij} \tag{1}$$

$$\text{s.t.} \sum_{(i,j)\in\delta^+(i)} x_{ij} = 1 \qquad \forall i \in C \tag{2}$$

$$\sum_{(j,i)\in\delta^-(i)} x_{ji} = \sum_{(i,j)\in\delta^+(i)} x_{ij} \qquad \forall i \in V \tag{3}$$

$$\sum_{(j,i)\in\delta^-(i)} (T^r_{ji} + \tau^r_{ji}x_{ji}) \leq \sum_{(i,j)\in\delta^+(i)} T^r_{ij} \qquad \forall r \in R,\ \forall i \in C \tag{4}$$

$$a_i x_{ij} \leq T^r_{ij} \leq b_i x_{ij} \qquad \forall r \in R,\ \forall (i,j) \in A \tag{5}$$

$$T^r_{ij} \geq 0 \qquad \forall r \in R,\ \forall (i,j) \in A \tag{6}$$

$$x_{ij} \in \{0,1\} \qquad \forall (i,j) \in A \tag{7}$$

The objective (1) sums up the cost of the used arcs. Constraints (2) ensure that each customer is visited exactly once, and (3) are the flow conservation constraints. Constraints (4) and (5) ensure the resource windows are satisfied. assumed that the bounds on the depot are always satisfied, since these can be reflected on the neighbouring nodes bounds. Note, no sub-tours can be present since only one time stamp per arc exists and the travel times are positive.

2 Bounded partial paths

A solution to the VRPTW: $v_0 \to c^1_1 \to \ldots \to c^1_{k_1} \to v_0, v_0 \to c^2_1 \to \ldots \to c^2_{k_2} \to v_0, \ldots, v_0 \to c^n_1 \to \ldots \to c^n_{k_n} \to v_0$ can be represented by the giant-tour representation of Christofides and Eilon [3]:

$$v_0 \to c^1_1 \to \ldots \to c^1_{k_1} \to v_0 \to c^2_1 \to \ldots \to c^2_{k_2} \to v_0 \ldots \to v_0 \to c^n_1 \to \ldots \to c^n_{k_n} \to v_0$$

which is one long path visiting all customers. The consumption of resources is reset each time the depot node is encountered.

The idea is to partition the problem so that the solution space of each part is smaller than the original problem. This is done by splitting the giant-tour into smaller segments by imposing an upper limit on some resource, e.g., bounding the path length in the number of nodes. In the following the number of visited customers is considered the bounding resource, i.e., the number of visits to the non-depot node set C. Each segment represents a partial path of the giant-tour. With a fixed number of customers on each partial path, say L, a fixed number of partial paths, say K, is needed to ensure that all customers are visited, i.e.,

$L \cdot K \geq |C|$. The partial paths can start and end in any node in V and can visit the depot several times. An example of a partial path is

$$c_1 \rightarrow c_2 \rightarrow v_0 \rightarrow c_3 \rightarrow v_0 \rightarrow c_4$$

Consider the graph $G'(V', A')$ consisting of a set of layers $K = \{1, \ldots, |K|\}$, each one representing G for a partial path. Let G^k be the sub graph of G' representing layer k with node set $V^k = \{(i, k) : i \in V\}$ for all $k \in K$ and arc set $A^k = \{(i, j, k) : (i, j) \in A\}$ for all $k \in K$. Let $A^* = \{(i, i, k) : (i, k) \in V^k \wedge (i, k+1) \in V^{k+1} \wedge k \in K\}$ be the set of interconnecting arcs, i.e., the arcs connecting a layer k with the layer above k namely layer $k+1$ for all $k \in K$ and all nodes $i \in V$ (layer $|K|+1$ is defined to be layer $1 \in K$ and layer 0 is defined to be layer $|K| \in K$). Let $V' = \bigcup_{k \in K} V^k$ and let $A' = \bigcup_{k \in K} A^k \cup A^*$. An illustration of G' can be seen on Figure 1. Note, that arc (i, i, k) does not exist in A^k and that arc (i, j, k) with $i \neq j$ does exist in A^*, so all arcs $(i, j, k) \in A'$ can be uniquely indexed. With the length of a path defined as the number of customers on it, the problem is now to find partial paths of length at most L in $|K|$ layers with $L \cdot |K| \geq |C| > L \cdot (|K| - 1)$, so that each partial path p ending in node $i \in V$ is met by another partial path p' starting in i. All partial paths are combined while not visiting any customers more than once and satisfying all resource windows. A customer $c \in C$ is considered to be on a partial path p if c is visited on p and is not the end node of p.

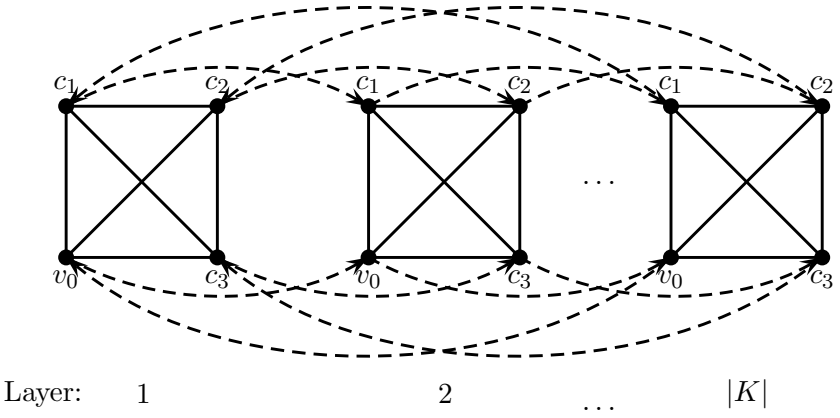


Figure 1: Illustration of G' with $|C| = 3$, $|K| = 3$, and $|L| = 1$. Edges (full-drawn) represent two arcs; one in each direction. Dashed lines are the interconnecting arcs A^*.

Let L be the upper bound on the length of each partial path, and let $|C|$ be the length of the combined path (the giant-tour). Now, exactly $|K| = \lceil |C|/L \rceil$ partial paths are needed to make the combined path, since $L \lceil |C|/L \rceil \geq |C| > L(\lceil |C|/L \rceil - 1)$. Note that given a $|K|$, L can be reduced to $L = \lceil |C|/|K| \rceil$.

3-index formulation of the VRPTW Let x_{ij}^k be the variable indicating the use of arc $(i,j,k) \in A'$. Problem (1)–(7) is rewritten:

$$\min \sum_{k\in K} \sum_{(i,j)\in A} c_{ij} x_{ij}^k \tag{8}$$

$$\texttt{s.t.} \sum_{k\in K} \sum_{(i,j)\in\delta^+(i)} x_{ij}^k = 1 \qquad \forall i \in C \tag{9}$$

$$\sum_{(i,j)\in\delta^+(i)} x_{ij}^k \leq 1 \qquad \forall k \in K,\ \forall i \in C \tag{10}$$

$$\sum_{k\in K} \left(x_{ii}^{k-1} + \sum_{(j,i)\in\delta^-(i)} x_{ji}^k \right) = \sum_{k\in K} \left(x_{ii}^k + \sum_{(i,j)\in\delta^+(i)} x_{ij}^k \right) \qquad \forall i \in V \tag{11}$$

$$x_{ii}^{k-1} + \sum_{(j,i)\in\delta^-(i)} x_{ji}^k = x_{ii}^k + \sum_{(i,j)\in\delta^+(i)} x_{ij}^k \qquad \forall k \in K,\ \forall i \in V \tag{12}$$

$$\sum_{k\in K} \sum_{i\in V} x_{ii}^k = K \tag{13}$$

$$\sum_{i\in C} \sum_{(i,j)\in A} x_{ij}^k \leq L \qquad \forall k \in K \tag{14}$$

$$\sum_{k\in K} \sum_{(j,i)\in\delta^-(i)} \left(T_{ji}^{rk} + \tau_{ji}^r x_{ji}^k \right) \leq \sum_{k\in K} \sum_{(i,j)\in\delta^+(i)} T_{ij}^{rk} \qquad \forall r \in R,\ \forall i \in C \tag{15}$$

$$\sum_{(j,i)\in\delta^-(i)} \left(T_{ji}^{rk} + \tau_{ji}^r x_{ji}^k \right) \leq \sum_{(i,j)\in\delta^+(i)} T_{ij}^{rk} \qquad \forall r \in R,\ \forall k \in K,\ \forall i \in C \tag{16}$$

$$a_i \sum_{k\in K} x_{ij}^k \leq \sum_{k\in K} T_{ij}^{rk} \leq b_i \sum_{k\in K} x_{ij}^k \qquad \forall r \in R,\ \forall (i,j) \in A \tag{17}$$

$$a_i x_{ij}^k \leq T_{ij}^{rk} \leq b_i x_{ij}^k \qquad \forall r \in R,\ \forall k \in K,\ \forall (i,j) \in A \tag{18}$$

$$x_{ij}^k \in \{0,1\} \qquad \forall k \in K,\ \forall (i,j) \in A \tag{19}$$

$$T_{ij}^{rk} \geq 0 \qquad \forall r \in R,\ \forall k \in K,\ \forall (i,j) \in A \tag{20}$$

The objective (8) sums up the cost of the used edges. Constraints (9) ensure that all customers are visited exactly once, while the redundant constraints (10) ensure that no customer is visited more than once. Constraints (11) maintain flow conservation between the original nodes V, and can be rewritten as

$$\sum_{k\in K} \sum_{(j,i)\in\delta^-(i)} x_{ji}^k = \sum_{k\in K} \sum_{(i,j)\in\delta^+(i)} x_{ij}^k \qquad \forall i \in V$$

since $\sum_{k\in K} x_{ii}^{k-1} = \sum_{k\in K} x_{ii}^k$. Constraints (12) maintain flow conservation within a layer. Constraint (13) ensures that K partial paths are selected and constraints (14) that the length of the partial path in each layer is at most L. Constraints (15) connect the resource variables on a global level and constraints (16) connect the resource variables within each single layer, note that since there is no (15) and (16) for the depot it is not constrained by resources. Constraints (17) globally enforce the resource windows and the redundant constraints (18) enforce the resource windows within each layer.

3 Dantzig-Wolfe decomposition

The 3-index formulation of the VRPTW (8)–(20) is Dantzig-Wolfe decomposed whereby a master and a pricing problem is obtained.

Master problem: Let λ_p be the variable indicating the use of partial path p. Using Dantzig-Wolfe decomposition where the constraints (9), (11), (13), (15), and (17) are kept in the master problem the following master problem is obtained:

$$\min \sum_{p \in P} c_p \lambda_p \tag{21}$$

$$\texttt{s.t.} \sum_{p \in P} \sum_{(i,j) \in \delta^+(i)} \alpha_{ij}^p \lambda_p = 1 \qquad \forall i \in C \tag{22}$$

$$\sum_{p \in P: e^p = i} \lambda_p = \sum_{p \in P: s^p = i} \lambda_p \qquad \forall i \in V \tag{23}$$

$$\sum_{p \in P} \lambda_p = K \tag{24}$$

$$\sum_{(j,i) \in \delta^-(i)} \left(T_{ji}^r + \sum_{p \in P} \tau_{ji}^r \alpha_{ji}^p \lambda_p \right) \leq \sum_{(i,j) \in \delta^+(i)} T_{ij}^r \qquad \forall r \in R,\ \forall i \in C \tag{25}$$

$$a_i \sum_{p \in P} \alpha_{ij}^p \lambda_p \leq T_{ij}^r \leq b_i \sum_{p \in P} \alpha_{ij}^p \lambda_p \qquad \forall r \in R,\ \forall (i,j) \in A \tag{26}$$

$$T_{ij}^r \geq 0 \qquad \forall r \in R,\ \forall (i,j) \in A \tag{27}$$

$$\lambda_p \in \{0,1\} \qquad \forall p \in P \tag{28}$$

Where α_{ij}^p is the number of times arc $(i,j) \in A$ is used on path $p \in P$ and s^p and e^p indicates the start and the end node respectively of partial path $p \in P$. Constraints (22) ensure that each customer is visited exactly once. Constraints (23) link the partial paths together by flow conservation. Constraint (24) is the convexity constraint ensuring that K partial paths are selected. Constraints (25) and (26) enforce the resource windows.

Pricing problem: The $|K|$ pricing problems corresponding to the master problem (21)–(28) contains constraints (10), (12), (14), (16), and (18) and can be formulated as a single ESPPRC where the depot is allowed to be visited more than once. Let s and e be a super source and a super target node respectively. Arcs (s,i) and (i,e) for all $i \in V$ are added to

G.

$$\min \sum_{(i,j)\in A} \overline{c}_{ij} x_{ij} \tag{29}$$

$$\text{s.t.} \sum_{(s,i)\in\delta^+(s)} x_{si} = 1 \tag{30}$$

$$\sum_{(i,e)\in\delta^-(e)} x_{ie} = 1 \tag{31}$$

$$\sum_{(i,j)\in A} x_{ij} \leq 1 \qquad \forall i \in C \tag{32}$$

$$\sum_{(j,i)\in\delta^-(i)} x_{ji} = \sum_{(i,j)\in\delta^+(i)} x_{ij} \qquad \forall i \in V \tag{33}$$

$$\sum_{i\in C} \sum_{(i,j)\in A} x_{ij} \leq L \tag{34}$$

$$\sum_{(j,i)\in\delta^-(i)} (T^r_{ji} + \tau^r_{ji} x_{ji}) \leq \sum_{(i,j)\in\delta^+(i)} T^r_{ij} \qquad \forall r \in R,\ \forall i \in C \tag{35}$$

$$a_i x_{ij} \leq T^r_{ij} \leq b_i x_{ij} \qquad \forall r \in R,\ \forall (i,j) \in A \tag{36}$$

$$x_{ij} \in \{0,1\} \qquad \forall (i,j) \in A \tag{37}$$

The objective (29) minimizes the reduced cost of a column. Constraints (30) and (31) ensure that the path starts in s respectively ends in e. Constraints (32) dictates that no node is visited more than once, thereby ensuring elementarity, and constraints (33) conserve the flow. Constraints (35) and (36) ensure the resource windows are satisfied for all customers. Note, since the depot is missing in (35) each time a path leaves the depot a resource is only restricted by its lower limit a^r_0 for all $r \in R$.

Let π ($\pi_i \geq 0 : \forall i \in C$) be the duals of (22), let $\pi_0 = 0$, let μ be the duals of (23), let $\beta \leq 0$ be the dual of (24), let ν ($\nu \leq 0 : \forall i \in C$) be the duals of (25), let $\nu_0 = 0$, and let $\underline{\omega} \leq 0$ and $\overline{\omega} \geq 0$ be the dual of (26). The cost of the arcs in this ESPPRC are then given as:

$$\overline{c}_{ij} = -\beta + \begin{cases} c_{ij} - \pi_i - \tau_{ij}\nu_j - a_i\underline{\omega}_i + b_i\overline{\omega}_i & \forall (i,j) \in A \setminus (\delta^+(s) \cup \delta^-(e)) \\ \mu_j & \forall (s,j) \in \delta^+(s) \\ \mu_i & \forall (i,e) \in \delta^-(e) \end{cases}$$

and the pricing problem becomes finding the shortest path from s to e.

Solving the pricing problem: The ESPPRCs can be solved by a labeling algorithm. For details regarding labeling algorithms we refer to Desaulniers et al. [4], Irnich [5], Irnich and Desaulniers [6], and Righini and Salani [9].

Branching: Integrality can be obtained by branching on the original variables, which can be accomplished by cuts in the master problem (see Vanderbeck [10]), e.g., let X_{ij} be the set of partial paths that utilize arc (i,j) then the branch rule $x_{ij} = 0 \ \vee \ x_{ij} = 1$ can be expressed by:

$$\sum_{p\in X_{ij}} \lambda_p = 0 \ \vee \ \sum_{p\in X_{ij}} \lambda_p = 1.$$

Bounds: The following theorem justifies the approach presented in this paper.

Theorem 1. Let z_{lp} be an LP-relaxed solution to (1)–(7) and let z_{pp} be an LP-relaxed solution to (21)–(28) then $Z_{lp} \leq Z_{pp}$ for all instances of the VRPTW and $Z_{lp} < Z_{pp}$ for some instances of the VRPTW.

Proof. $Z_{lp} \leq Z_{pp}$ since all solutions to (21)–(28) map to solutions to (1)–(7). An instance with $Z_{lp} < Z_{pp}$ is obtained with four customers each with a demand of resource r of half the global maximum b_r of r, the distance from the customers to the depot larger than the distance between the customers, and $L = 4$. The solution to (21)–(28) would use the expensive edges four times, whereas the solution to (1)–(7) only would use them twice. □

4 Conclusion

A new decomposition model of the VRPTW has been presented with ESPPRCs as the pricing problems. The model facilitates control of the running time of the pricing problems. Due to the aggregation of the model, LP relaxed bounds of (21)–(28) are better than the direct model (1)–(7). Since (21)–(28) is a relaxation of the traditional Dantzig-Wolfe decomposition model with elementary routes as columns, the LP relaxed bounds may be weaker yielding a larger branch-and-bound tree. The difference in bound quality can be decreased with the use of special purpose cutting planes, which this paper does not leave room for. Furthermore, effective cuts such as Subset Row-inequalities by Jepsen et al. [7] and Chvátal-Gomory Rank-1 cuts (see Petersen et al. [8]) can be applied to the Set Partition master problem to strengthen the bound. Future experimental results will conclude on the effectiveness of this approach.

References

[1] N. Ascheuer, M. Fischetti, and M. GrÃ¶tschel. Solving the asymmetric travelling salesman problem with time windows by branch-and-cut. *Mathematical Programming*, 90(3): 475–506, 2001. doi: 10.1007/PL00011432.

[2] J. F. Bard, G. Kontoravdis, and G. Yu. A branch-and-cut procedure for the vehicle routing problem with time windows. *Transportation Science*, 36(2):250–269, May 2002. doi: http://dx.doi.org/10.1287/trsc.36.2.250.565.

[3] N. Christofides and S. Eilon. An algorithm for the vehicle-dispatching problem. *Operational Research Quarterly*, 20(3):309–318, Sep 1969.

[4] G. Desaulniers, J. Desrosiers, J. Ioachim, I. M. Solomon, F. Soumis, and D. Villeneuve. A unified framework for deterministic time constrained vehicle routing and crew scheduling problems. In T. G. Crainic and G. Laporte, editors, *Fleet Management and Logistics*, pages 57–93. Kluwer, 1998.

[5] S. Irnich. Resource extension functions: Properties, inversion, and generalization to segments. *OR Spectrum*, 30(1):113–148, 2008. doi: 10.1007/s00291-007-0083-6.

[6] S. Irnich and G. Desaulniers. Shortest path problems with resource constraints. In G. Desaulniers, Jacques Desrosiers, and M.M. Solomon, editors, *Column Generation*, chapter 2, pages 33–65. Springer, 2005. doi: 10.1007/0-387-25486-2_2.

[7] M. Jepsen, B. Petersen, and S. Spoorendonk. A branch-and-cut algorithm for the elementary shortest path problem with a capacity constraint. Technical Report 08/01, DIKU Department of Computer Science, University of Copenhagen, Denmark, 2008.

[8] B. Petersen, D. Pisinger, and S. Spoorendonk. Chvátal-Gomory rank-1 cuts used in a Dantzig-Wolfe decomposition of the vehicle routing problem with time windows. In B. Golden, R. Raghavan, and E. Wasil, editors, *The Vehicle Routing Problem: Latest Advances and New Challenges*, pages 397–420. Springer, 2008. doi: 10.1007/978-0-387-77778-8_18.

[9] G. Righini and M. Salani. Symmetry helps: bounded bi-directional dynamic programming for the elementary shortest path problem with resource constraints. *Discrete Optimization*, 3(3):255–273, 2006. doi: 10.1016/j.disopt.2006.05.007.

[10] F. Vanderbeck. On Dantzig-Wolfe decomposition in integer programming and ways to perform branching in a branch-and-price algorithm. *Operation Research*, 48(1):111–128, 2000. doi: 10.1287/opre.48.1.111.12453.

Chapter 14

The Vehicle Routing Problem Solved by Bounding and Enumeration of Partial Paths

Mads Jepsen
DTU Management Engineering, Technical University of Denmark

Bjørn Petersen
DTU Management Engineering, Technical University of Denmark

1 Introduction

The Capacitated Vehicle Routing Problem (CVRP) can be described as follows: A set of customers, each with a demand, needs to be serviced by a number of vehicles all starting and ending at a central depot. Each customer must be visited exactly once and the capacity of the vehicles may not be exceeded. The objective is to service all customers traveling the least possible distance. In this abstract we consider a homogeneous fleet, i.e., all vehicles are identical. The Vehicle Routing Problem with Time Windows (VRPTW) extends the CVRP by imposing that each customer must be visited within a given time window. The intersection of the CVRP and the VRPTW will in the following be referred to as the VRP.

The standard Dantzig-Wolfe decomposition of the arc flow formulation of the VRP is to split the problem into a master problem (a Set Partitioning Problem) and a pricing problem (an Elementary Shortest Path Problem with Resource Constraints (ESPPRC), where capacity (and time) are the constrained resources). A restricted master problem can be solved with delayed column generation and embedded in a branch-and-bound algorithm to ensure integrality. Applying cutting planes either in the master or the pricing problem leads to a Branch-and-Cut-and-Price algorithm (BCP). Kohl et al. [6] implemented a successful BCP algorithm for the VRPTW by applying *sub-tour elimination* constraints and *two-path* cuts, Cook and Rich [2] generalized the *two-path* cuts to the *k-path* cuts, and Fukasawa et al. [3]

Extended abstract, Tristan 2010.

applied a range of valid inequalities for the CVRP based on the branch and cut algorithm of Lysgaard et al. [7]. Common for these BCP algorithms is that all applied cuts are valid inequalities for the VRPTW respectively the CVRP with regard to the *original* arc flow formulation, and have a structure which makes it possible to handle values of the dual variables in the pricing problem without increasing the complexity of the problem. The BCP algorithm was extended to include valid inequalities for the master problem by applying the subset row (SR) inequalities to the Set Partitioning master problem in Jepsen et al. [5] and later by applying Chvátal-Gomory Rank-1 (CG1) inequalities in Petersen et al. [8]. Baldacci et al. [1] use an approach where columns with potentially negative reduced cost are enumerated after initial upper and lower bounds are found, this sometimes leads to memory issues with difficult instances. After enumeration a general MIP solver is called. Recently, Jepsen and Petersen [4] presented an new decomposition model based on bounded partial paths, where the solution space of the pricing problem is limited by bounding some resource.

We propose to combine the latter two strategies, i.e., enumeration of columns with potentially negative reduced with the columns being bounded partial paths. The main ideas of Baldacci et al. [1] would be utilised until the enumeration step where the partial path columns would be used instead of the much larger set of elementary routes, thus hopefully solving the memory issues notied in Baldacci et al. [1]. The gap between the lower (LB) and the upper bound (UB) of the master problem obtained with the elementary routes can be maintained by bounding LB.

2 Mathematical Model

The VRP can formally be stated as: Given a graph $G(V, A)$ with nodes V and arcs A, a set R of resources $R = \{\text{load (and time)}\}$ where each resource $r \in R$ has a lower bound a_i^r and an upper bound b_i^r for all $i \in V$ and a positive consumption τ_{ij}^r when using arc $(i, j) \in A : i \in C$, find a set of routes starting and ending at the depot node $0 \in V$ satisfying all resource limits, such that the cost is minimized and all customers $C = V \setminus \{0\}$ are visited. The *load* resource is present for both the CVRP and the VRPTW and ensures that the capacity of the vehicle is not violated when satisfying the demand of the customers. The *time* resource is only present for the VRPTW and ensures that the customers are serviced during a predifined time window.

In the following let c_p be the cost of partial path $p \in P$, λ_p be the binary variable indicating the use of p, and T_{ij}^r (the resource stamp) be the consumption of resource $r \in R$ at the beginning of arc $(i, j) \in A$. Let $\delta^+(i)$ and $\delta^-(i)$ be the set of outgoing respectively ingoing arcs of node $i \in V$. Finally, let LB be a given lower bound. The master problem:

$$\min \sum_{p \in P} c_p \lambda_p \tag{1}$$

$$\text{s.t.} \sum_{p \in P} c_p \lambda_p \geq LB \tag{2}$$

$$\sum_{p \in P} \sum_{(i,j) \in \delta^+(i)} \alpha_{ij}^p \lambda_p = 1 \qquad \forall i \in C \tag{3}$$

$$\sum_{p \in P : e^p = i} \lambda_p = \sum_{p \in P : s^p = i} \lambda_p \qquad \forall i \in V \tag{4}$$

$$\sum_{p \in P} \lambda_p = K \tag{5}$$

$$\sum_{(j,i) \in \delta^-(i)} \left(T^r_{ji} + \sum_{p \in P} \tau^r_{ji} \alpha^p_{ji} \lambda_p \right) \leq \sum_{(i,j) \in \delta^+(i)} T^r_{ij} \qquad \forall r \in R,\ \forall i \in C \tag{6}$$

$$a_i \sum_{p \in P} \alpha^p_{ij} \lambda_p \leq T^r_{ij} \leq b_i \sum_{p \in P} \alpha^p_{ij} \lambda_p \qquad \forall r \in R,\ \forall (i,j) \in A \tag{7}$$

$$T^r_{ij} \geq 0 \qquad \forall r \in R,\ \forall (i,j) \in A \tag{8}$$

$$\lambda_p \in \{0,1\} \qquad \forall p \in P \tag{9}$$

Where α^p_{ij} indicates the use arc $(i,j) \in A$ is used on path $p \in P$ and s^p and e^p indicate the start respectively the end node of partial path $p \in P$. Constraints (3) ensure that each customer is visited exactly once. Constraints (4) link the partial paths together by flow conservation. Constraint (5) is the convexity constraint ensuring that K partial paths are selected. Constraints (6) and (7) enforce the resource windows.

3 Algorithmic Overview

The algorithm is inspired by the one of Baldacci et al. [1]. Heuristics can be applied where ever being beneficial. The algortithm is divided into the follwoing steps:

i By the use of column generation with columns being elementary routes and all advantageous cuts being utilized (e.g., (SR)), find an initial good quality LB and initial good quality UB.

ii Solve the LP-relaxed master problem (1)–(9) with the LB from i.

iii Due to the bounds (9) each column in P cannot be in a solution more than once, hence, any partial path $p \in P$ in an optimal solution must satisfy: $\bar{c}_p \leq UB - LB$, where $\bar{c}_p$ is the reduced cost of column p in the last iteration of ii. Enumerate all these.

iv Apply all the columns from iii to the master problem, add the cuts of Section 4, and give the problem to a general MIP-solver.

4 Tightening Bounds

Constraints (6) and (7) can be tightened by:

$$\sum_{p \in P : e^p = i} (a_p + \tau_p) \lambda_p \leq \sum_{(i,j) \in \delta^+(i)} T_{ij} \qquad \forall i \in C \tag{10}$$

$$\sum_{p \in P, s^p = i} a_p \lambda_p \leq \sum_{(i,j) \in \delta^+(i)} T_{ij} \leq \sum_{p \in P, s^p = i} b_p \lambda_p \qquad \forall i \in V \tag{11}$$

where a_p, b_p, and τ_p are bounds on the partial path p and due to integrality on p can yield tighter bounds. Lower bound a_p for p is defined as the latest possible departure time from the start-node s without changing the earliest possible arrival time at the end-node e. Upper bound b_p for p is defined as the latest possible departure time from s while p still being feasible.

Travel time τ_p is defined as the time spend on p, i.e., traversing edges and waiting for windows to open. It is noted that τ_p is always constant given a_p and b_p as defined above no matter which departure time $t : a_p \leq t \leq b_p$, since the traversal times of edges are constant and a difference in waiting time would yield a conflict with the definition of a_p. As a consequence of this, $a_p = b_p$ if there is waiting time on p, and if $a_p \neq b_p$ then no waiting time occurs on p.

Even though the influence on the reduced cost with these constraints can be handled in the pricing problem, experience points to it being cumbersome to implement and having a negatively influencing on the running time of the pricing problem. In the context of enumeration the dual cost of these constraints do not have to be handled since they are added after the enumeration procedure, thus obtaining the smaller solution space for "free".

References

[1] R. Baldacci, N. Christofides, and A. Mingozzi. An exact algorithm for the vehicle routing problem based on the set partitioning formulation with additional cuts. *Mathematical Programming*, 115(2):351–385, 2008. doi: 10.1007/s10107-007-0178-5.

[2] W. Cook and J. L. Rich. A parallel cutting plane algorithm for the vehicle routing problem with time windows. Technical Report TR99-04, Computational and Applied Mathematics, Rice University, Houston, Texas, USA, 1999.

[3] R. Fukasawa, H. Longo, J. Lysgaard, M. Poggi de AragÃ£o, M. Reis, E. Uchoa, and R.F. Werneck. Robust branch-and-cut-and-price for the capacitated vehicle routing problem. *Mathematical Programming*, 106(3):491–511, 2006. doi: 10.1007/s10107-005-0644-x.

[4] M. Jepsen and B. Petersen. Partial path column generation for the vehicle routing problem. Technical report, DTU Management Engineering, Technical University of Denmark, Produktionstorvet 424, 2800 Kgs. Lyngby, Denmark, 2009.

[5] M. Jepsen, B. Petersen, S. Spoorendonk, and D. Pisinger. Subset-row inequalities applied to the vehicle-routing problem with time windows. *Operations Research*, 56(2):497–511, 2008. doi: 10.1287/opre.1070.0449.

[6] N. Kohl, J. Desrosiers, O. B. G. Madsen, M. M. Solomon, and F. Soumis. 2-path cuts for the vehicle routing problem with time windows. *Transportation Science*, 33(1):101–116, 1999. doi: 10.1287/trsc.33.1.101.

[7] J. Lysgaard, A. N. Letchford, and R. W. Eglese. A new branch-and-cut algorithm for the capacitated vehicle problem. *Mathematical Programming*, 100(2):423–445, 2004. doi: 10.1007/s10107-003-0481-8.

[8] B. Petersen, D. Pisinger, and S. Spoorendonk. Chvátal-Gomory rank-1 cuts used in a Dantzig-Wolfe decomposition of the vehicle routing problem with time windows. In B. Golden, R. Raghavan, and E. Wasil, editors, *The Vehicle Routing Problem: Latest Advances and New Challenges*, pages 397–420. Springer, 2008. doi: 10.1007/978-0-387-77778-8_18.

Chapter 15

A solution approach to the ROADEF/EURO 2010 challenge based on Benders Decomposition

Richard Lusby
DTU Management Engineering, Technical University of Denmark

Laurent F. Muller
DTU Management Engineering, Technical University of Denmark

Bjørn Petersen
DTU Management Engineering, Technical University of Denmark

Abstract

Since 1999 the French Operations Research Society, Recherche Opérationnelle et d'Aide à la Décision (ROADEF), has organized the so-called ROADEF challenges, an international operations research contest in which participants must solve an industrial optimization problem. In 2010 it was jointly organized for the first time with the European Operational Research Society (EURO) and was run in collaboration with Electricité de France (EDF), one of the largest utility companies in the world, and required contestants to solve a large scale energy management problem with varied constraints. The challenge focused on the nuclear power plants, which need to be regularly shut down for refueling and maintenance, and asked contestants to schedule these outages such that the expected cost of meeting the power demand in a number of potential scenarios was minimized.

We present a Benders decomposition based framework for solving the problem. Because of the nature of the problem, not all constraints can be modeled satisfactory as linear constraints and the approach is therefor divided into two stages: in the first stage Benders feasibility and optimality cuts are added based on the linear programming relaxation of the Benders Master problem, and in the second stage feasible integer solutions are enumerated and procedure is applied to each solution in an attempt to make them satisfy the constraints not part of the mixed integer program. A number of experiments are performed on the available benchmark instances. These experiments show that the

Technical report.

approach is competitive on the smaller instances, but not for the larger ones. We believe the exact approach gives insight into the problem, and additionally makes it possible to find lower bounds on the problem, which is typically not the case for the competing heuristics.

Keywords: Benders decomposition, power scheduling

1 Introduction

Every two years[1] since 1999 the French Operations Research Society, Recherche Opérationnelle et d'Aide à la Décision (ROADEF), has organized the so-called ROADEF challenge, an international operations research contest in which participants must solve an industrial optimization problem. Given the success of previous contests, this year it was jointly organized for the first time with the European Operational Research Society (EURO) and known as the ROADEF/EURO 2010 challenge. The competition was run in collaboration with Electricité de France (EDF), one of the largest utility companies in the world, and required contestants to solve a large scale energy management problem with varied constraints.

EDF's power generation facilities in France stand for a total of 98.8 GW of installed capacity, most of which is produced using thermal, and in particular nuclear, power plants. In 2008 thermal power plants accounted for 90% of its total electricity production of which 86% was delivered by nuclear power plants. This year's challenge focused on the nuclear power plants, since these need to be regularly shut down for refueling and maintenance, and asked contestants to schedule these outages in such a way that the various constraints regarding safety, maintenance, logistics, and plant operation were satisfied, while minimizing the expected cost of meeting the power demand in a number of potential scenarios. The problem thus consisted of the following two dependent subproblems

1. Determine a schedule of nuclear power plant outages. This entails determining when the nuclear power plants should be taken *offline* and how much fuel should be reloaded at each. An outage lasts for some predefined (plant specific) period of time during which the nuclear power plant cannot be used for power generation. The coupling of an outage followed by a production period (until the next outage) for a nuclear power plant is termed a *cycle* and it is not uncommon to have to schedule up to six cycles for each nuclear power plant. In determining an outage schedule one must obey several safety requirements as well as observe restrictions arising from the limited resources available to perform the fuel reloading.

2. Given an outage schedule, determine a production plan for each of the *online* power plants, i.e. the quantity of electricity to produce in each time step, for each possible demand scenario. The power plants are divided into two categories termed Type 1 and Type 2, respectively. Type 2 power plants refer to the nuclear power plants and must be reloaded with fuel, while Type 1 power plants represents thermal power plants, which can be supplied with fuel continuously, such as coal, gas, and oil powered plants. Several technical constraints govern the possible levels of power production at each power plant. Due to the stochastic nature of power markets, one is required to consider multiple demand scenarios.

[1] Except for 2009-2010.

The concepts of cycles, outages, and production plans for three power plants are illustrated in Figure 1. The gray area indicates the time steps during which the plants are offline.

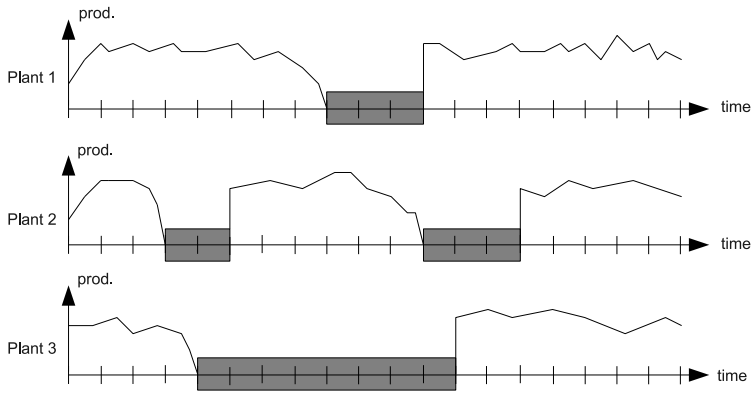


Figure 1: Outages, Cycles, Production Plans

One important aspect of this years problem was its size: There were approximately one hundred power plants and scenarios, and the planning horizon was in the order of years, with a granularity down to hours. Which means that a solution alone can contain in the order of 10^8 variables.

The remaining of the paper is organized as follows: Section 2 gives an overview of the problem constraints, Section 3 presents a mixed integer programming (MIP) model for (parts of) the problem, Section 4 gives a general outline of the proposed procedure and presents the Benders decomposed model, Section 5 describes how the problem size can be reduced, Section 6 describes a number of additional constraints that are added to the model in an attempt to reduce the number of infeasible subproblems, Section 7 describes a procedure for taking a solution, which does not satisfy all the constraints, and making it do so, Section 8 presents the computational results, and finally we conclude in Section 9.

2 Overview of problem constraints

Table 1 give a brief overview of the different constraints of the problem. Because of space considerations, we do not include a full description of the constraints, but instead refer the reader to Porcheron et al. [8]. We first introduce a number of sets, constants, which will also be used when stating the mathematical model in Section 3, and some additional sets and variables, which will only be used to describe the constraints in this section.

Sets

- I: Set of type 2 plants.
- J: Set of type 1 plants.
- T: Set of time steps.
- W: Set of weeks.

- S: Set of scenarios.
- K_i: Set of cycles for each plant $i \in I$.

Constants

- L_{ik}: Length in weeks of the outage for cycle $k \in K_i$ at plant $i \in I$.
- $\underline{R}_{ik}$: Minimum reload amount for plant $i \in I$ in cycle $k \in K_i$.
- $\overline{R}_{ik}$: Maximum reload amount for plant $i \in I$ in cycle $k \in K_i$.
- P_{it}^{max}: Maximum production for plant $i \in I$ at time step $t \in T$
- F_t: Conversion factor between power and fuel in time step $t \in T$.
- D_{ts}: Required power in time step $t \in T$ of scenario $s \in S$.
- B_{ik}: Fuel stock level at which shutdown curve must begin in cycle k at plant $i \in I$.
- Q_{ik}: Proportion of fuel that can be kept during reload in cycle k at plant $i \in I$, $\tilde{Q}_{ik} := \frac{Q_{ik}-1}{Q_{ik}}$.
- S_{ik}^{max}: Maximum permitted fuel after reload in cycle k at plant $i \in I$, $M_i := \max_k S_{i,k}^{max}$.
- A_{ik}^{max}: Maximum permitted fuel prior to reload in cycle k at plant $i \in I$.
- $\overline{P}_{itk}$: Maximum production capacity in cycle k for plant $i \in I$ at time step $t \in T$.
- $\overline{P}_{jts}$: Maximum production capacity for plant $j \in J$ at time step $t \in T$ in scenario $s \in S$.
- $\underline{P}_{jts}$: Minimum production capacity for plant $j \in J$ at time step $t \in T$ in scenario $s \in S$.
- X_i: Starting stock of plant $i \in I$
- T_{ik}^{O}: First possible outage week for cycle $k \in K_i$ for $i \in I$.
- T_{ik}^{A}: Last possible outage week for cycle $k \in K_i$ for $i \in I$.

Constraint-specific variables and sets

- $ha(i,k)$: the first week of the outage of cycle $k \in K_i$ of plant $i \in I$.
- $p(j,t,s)$: production of plant $j \in J$ during the time step $t \in T$ of scenario $s \in S$.
- $p(i,t,s)$: production of plant $i \in I$ during the time step $t \in T$ of scenario $s \in S$.
- $r(i,k)$: reload performed during the outage of cycle $k \in K_i$ of plant $i \in I$.
- $x(i,t,s)$: stock of fuel of plant $i \in I$ at time step $t \in T$ for scenario $s \in S$.
- $ec(i,k)$: set of time steps composing the production campaign of cycle $k \in K_i$ of plant $i \in I$.
- $ea(i,k)$: set of weeks composing the outage of cycle $k \in K_i$ of plant $i \in I$.

Table 1: Overview of the constraints of the problem

Name	Description
CT1	Constraints coupling load and production: during every time step $t \in T$ of every scenario $s \in S$, the sum of production of Type 1 and Type 2 power plants has to be equal to the demand: $$\sum_{i \in I} p_{its} + \sum_{j \in J} p_{jts} = D_{ts}, \quad \forall (t,s)$$
CT2	Bound on production: During every time step $t \in T$ of every scenario $s \in S$, production of plant $j \in J$ has to be between minimum and maximum bounds: $$\underline{P}_{jts} \leq p_{jts} \leq \overline{P}_{jts}, \quad \forall (j,t,s)$$
CT3	Offline power: During every time step $t \in T$ of every scenario $s \in S$ where plant $i \in I$ is on outage, its production is equal to zero: $$t \in ea(i,k) \Rightarrow p(i,t,s) = 0, \quad \forall (i,t,s)$$
CT4	Minimum power: During every time step $t \in T$ of every scenario $s \in S$ where plant $i \in I$ is online, its production is positive or equal to zero: $$0 \leq p(i,t,s), \quad \forall (i,t)$$
CT5	Maximum power before activation of imposition of power profile constraint (see CT6): During every scenario $s \in S$ and every time step $t \in T$ of the production campaign of cycle $k \in K_i$, if the current fuel stock of plant $i \in I$ is greater than or equal to B_{ik}, the production level has to be equal or less than its maximum bound: $$t \in ec(i,k) \wedge x(i,t,s) \geq B_{ik} \Rightarrow p(i,t,s) \leq \overline{P}_{it}, \quad \forall (i,t,s)$$
CT6	Maximum power after activation of imposition of power level constraint: During every scenario $s \in S$ and every time step $t \in T$ of the production campaign of cycle $k \in K_i$, if the current fuel stock of plant $i \in I$ is inferior to B_{ik}, production has to follow the power profile $\mathcal{P}_{ik} : \mathrm{R} \rightarrow [0;1]$ with a tolerance ϵ, where $\mathcal{P}_{ik}$ is a piecewise linear function of the stock level: $$t \in ec(i,k) \wedge x(i,t,s) \leq B_{ik} \Rightarrow p(i,t,s) \approx \mathcal{P}_{ik}(x(i,t,s)), \quad \forall (i,t,k,s)$$

Table 1: Continued – Overview of the constraints of the problem

Name	Description
CT7	Bounds on refueling: The reload performed during cycle $k \in K_i$ of plant $i \in I$ has to be inside its minimum and maximum bounds: $$\underline{R}_{ik} \leq r(i,k) \leq \overline{R}_{ik}, \quad \forall (i,k)$$
CT8	Initial fuel stock: $$x(i,0,s) = X_i, \quad \forall (i,s)$$
CT9	Fuel stock variation during a production campaign: $$t \in ec(i,k) \Rightarrow x(i,t+1,s) = x(i,t,s) - p(i,t,s) \cdot F_t, \quad \forall (t,i,k,s)$$
CT10	Fuel stock variation during an outage: In the process of refueling a Type 2 power plant at time $t \in T$, a certain amount of unspent fuel has to be removed to make the addition of new fuel possible: $$x(i,t+1,s) = \tilde{Q}_{ik} \cdot (x(i,t,s) - B_{i,k-1}) + r(i,k) + B_{ik}, \quad \forall (i,k,s)$$
CT11	Bounds on fuel stock at the instant, $t \in T$, of outage and after refueling: $$x(i,t,s) \leq A_{ik}^{max}, \quad x(i,t+1,s) \leq S_{ik}^{max}, \quad \forall (i,k,s)$$
CT12	Constraint on maximum modulation over a cycle: Every modulation of the power output of a Type 2 power plant leads to a certain amount of wear of the equipment involved. Therefore frequent power modulations at Type 2 power plants are undesirable: $$\sum_{t \in \{t' \in ec(i,k) : x(i,t',s) \geq B_{ik}\}} (\overline{P}_{it} - p(i,t,s)) \cdot F_t \leq M_{ik}^{max}, \forall (i,k,s)$$

Table 1: Continued – Overview of the constraints of the problem

Name	Description
CT13	Constraint on the date of outage at the earliest and the latest: Outage of cycle $k \in K_i$ of plant $i \in K_i$ has to start during a given interval: $T^O_{ik} \leq h(i,k) \leq T^A_{ik}, \quad \forall (i,k),$ $ha(i,k+1) \geq ha(i,k) + L_{ik}, \quad \forall (i,k)$ If no CT13 constraint is present, then scheduling the corresponding cycle is optional, but the cycle must still be scheduled in order for any subsequent cycle $k' > k$ to be scheduled.
CT14	Constraints on the minimum spacing/maximum overlapping between outages: Outages of a set A_m have to be spaced by at least S_m weeks, with $m = 1, \dots M_{14}$: $ha(i,k) - ha(i',k') - L_{i'k'} \geq S_m \vee ha(i',k') - ha(ik) - L_{ik} \geq S_m, \quad \forall (i,k),(i',k') \in A_m$
CT15	Minimum spacing/maximum overlapping between outages during a specific period: Outages of a set A_m that intersect an interval $[A_m; B_m]$ have to be spaced by at least or can overlap by at most S_m weeks, with $m = 1, \dots M_{15}$: $A_m - L_{ik} + 1 \leq ha(i,k) \leq B_m \wedge A_m - L_{i'k'} + 1 \leq ha(i',k') \leq B_m$ $\Rightarrow ha(i,k) - ha(i',k') - L_{i'k'} \geq S_m \vee ha(i',k') - ha(ik) - L_{ik} \geq S_m,$ $\forall (i,k),(i',k') \in A_m$
CT16	Minimum spacing constraint between decoupling dates: Dates of decoupling of outages of a set A_m have to be spaced by at least S_m weeks, with $m = 1, \dots M_{16}$: $\lvert ha(i,k) - ha(i',k') \rvert \geq S_m, \quad \forall (i,k),(i',k') \in A_m$
CT17	Minimum spacing constraints between dates of coupling: Coupling dates of outages of a set A_m have to be spaced by at least S_m weeks, with $m = 1, M_{17}$: $\lvert ha(i,k) + L_{ik} - ha(i',k') - L_{i'k'} \rvert \geq S_m, \quad \forall (i,k),(i',k') \in A_m$

Table 1: Continued – Overview of the constraints of the problem

Name	Description
CT18	Minimum spacing constraints between coupling and decoupling dates: Dates of coupling and decoupling of outages of a set A_m have to be spaced by at least S_m weeks, with $m = 1, \dots M_{18}$: $$\lvert ha(i,k) + L_{ik} - ha(i',k') \rvert \geq S_m, \quad \forall (i,k),(i',k') \in A_m$$
CT19	Resource constraints: Use of resources on a given set of outages A_m is subject to constraints due to their limited availability, with A_{ikm}, and B_{ikm} indicating the start and the length of the resource usage period with $m = 1, \dots M_{19}$: $$\sum_{(i,k) \in A_m} \delta(t,i,k) \leq Q_m, \quad \forall w,$$ where $\delta(t,i,k) = 1 \iff t \in [ha(i,k) + A_{ikm}; ha(i,k) + A_{ikm} + B_{ikm}]$
CT20	Constraint on the maximum number of overlapping outages during a given week: At most $N_m(w)$ outages of $A_m(w)$ can overlap during the week$w \in W$, with $m = 1, \dots M_{20}$: $$\sum_{(i,k) \in A_m} \delta(t,i,k) \leq N_m(w), \quad \forall w,$$ where $\delta(t,i,k) = 1 \iff t \in [ha(i,k); ha(i,k) + L_{ik}]$.
CT21	Constraint on the offline power capacity of a set of power plants during a time period: For a given period, $[A_m; B_m]$ the power capacity of the set of plants A_m that are on outage has to be inferior to a maximum bound, I_m^{max}, with $m = 1, \dots, M_{21}$: $$\sum_{i \in C_m} \sum_{w \in [A_m;B_m] \cap ec(i,k)} \sum_{t \in w} \overline{P}_{it} \leq I_m^{max}$$

3 Model

As the approach to solving the problem will be based on applying mixed integer programming (MIP) we now give a MIP model of the problem. Before stating the model we introduce some additional sets, constants, and variables.

Sets

- W_{ik}^o: Set of allowed outage weeks for cycle $k \in K_i$ of plant $i \in I$.
- W_{ik}^p: Set of weeks where cycle $k \in K_i$ of plant $i \in I$ could be in a production campaign.

- T^p_{ik}: Set of time steps where cycle $k \in K_i$ of plant $i \in I$ could be in a production campaign.
- $K_i(w)$: Set of cycles for plant $i \in I$ which could be in a production campaign at in week w.
- $w(t)$: Week containing time step t
- w_t: Set of time steps in week w
- M_{21}: Set of CT21 constraints
- C_m: Set of type 2 power plants associated with $m \in M_{21}$

Constants

- c_{jt}: Cost of producing a unit of power at plant $j \in J$ in time step $t \in T$.
- $c_{i,|T|+1}$: Price of remaining fuel (time step $|T|+1$) at plant $i \in I$.

Variables

- y_{iwk}: Binary variable indicating if cycle k for plant i begins in week $w \in W^o_{ik}$
- r_{ik}: The amount of fuel reloaded in cycle k for plant i
- x^b_{iks}: Stock at the beginning of cycle k for plant $i \in I$ in scenario $s \in S$.
- x^e_{iks}: Stock at end of cycle k for plant $i \in I$ in scenario $s \in S$.
- x^f_{is}: Final stock for plant $i \in I$ in scenario $s \in S$.
- p_{itks}: Amount of power produced at plant $i \in I$ in cycle k at time step $t \in T$ in scenario $s \in S$.
- p_{jts}: Amount of power produced at plant $j \in J$ in time step $t \in T$ in scenario $s \in S$.
- $$\rho(i,w,k) := \begin{cases} 1 - \sum_{w' \le w} y_{i,w',k+1}, & k = 0 \\ \sum_{w' \le w - L_{ik}} y_{i,w',k}, & k = K \\ \sum_{w' \le w - L_{ik}} y_{i,w',k} - \sum_{w' \le w} y_{i,w',k+1}, & \text{otherwise} \end{cases}$$

  Note that $\rho(i,w,k)$ is not a variable, but is included for ease of exposition. The following relation holds $\rho(i,w,k) = 1 \iff$ cycle (i,k) is in a production campaign in week w.

Model

$$\min \sum_{i\in I}\sum_{k\in K} c_{ik}r_{ik} + \frac{1}{|S|}\sum_{s\in S}(\sum_{t\in T}\sum_{j\in J} c_{jts}F_t p_{jts} - \sum_{i\in I} c_{i,|T|+1}x^f_{is}) \tag{1}$$

$$\text{s.t. } r_{ik} \geq \underline{R}_{ik} \cdot \sum_{w\in W_{ik}} y_{iwk} \qquad \forall i\in I, \forall k\in K_i \tag{2}$$

$$r_{ik} \leq \overline{R}_{ik} \cdot \sum_{w\in W_{ik}} y_{iwk} \qquad \forall i\in I, \forall k\in K_i \tag{3}$$

$$\sum_{w\in W_{ik}} y_{iwk} \geq \sum_{w\in W_{i,k+1}} y_{i,w,k+1} \qquad \forall i\in I, \forall k\in K_i \tag{4}$$

$$\sum_{i\in C_m}\sum_{k\in K_i}\sum_{w\in IT_m}\sum_{w'=w-L_{ik}+1}^{w} y_{iw'k} \cdot \sum_{t=t(w)}^{t(w+1)-1} P^{max}_{it} \leq I^{max}_m \qquad \forall m\in M_{21}, \forall w\in W \tag{5}$$

$$\sum_{(iwk)\in H} y_{iwk} \leq K_H \qquad \forall H\in\mathcal{H} \tag{6}$$

$$x^e_{iks} = x^b_{iks} - \sum_{t\in T} p_{itks}\cdot F_t \qquad \forall i\in I, \forall k\in K_i, \forall s\in S \tag{7}$$

$$x^b_{iks} = r_{ik} + B_{ik}\sum_{w\in W^o_{ik}} y_{iwk} + \tilde{Q}_{ik}\left(x^e_{i,k-1,s} - B_{i,k-1}\sum_{w\in W^o_{ik}} y_{iwk}\right) \qquad \forall i\in I, \forall k\in K_i, \forall s\in S \tag{8}$$

$$x^e_{iks} \leq A^{max}_{i,k+1} + \left(1 - \sum_{w\in W^o_{ik}} y_{i,w,k+1}\right)\left(M^1_i - A^{max}_{i,k+1}\right) \qquad \forall i\in I, \forall k\in K_i, \forall s\in S \tag{9}$$

$$x^b_{iks} \leq S^{max}_{ik} + \left(1 - \sum_{w\in W^o_{ik}} y_{iwk}\right)(M_i - S^{max}_{ik}) \qquad \forall i\in I, \forall k\in K_i, \forall s\in S \tag{10}$$

$$x^f_{is} \leq \sum_{k'>k}\sum_{w\in W^o_{ik}} y_{iwk'}M_i + x^e_{iks} \qquad \forall i\in I, \forall k\in K_i, \forall s\in S \tag{11}$$

$$p_{itks} \leq \overline{P}_{it}\cdot\rho(i,w(t),k) \qquad \forall i\in I, \forall k\in K_i, \forall t\in T^p_{ik}, \forall s \tag{12}$$

$$\underline{P}_{jts} \leq p_{jts} \leq \overline{P}_{jts} \qquad \forall j\in J, \forall t\in T, \forall s\in S \tag{13}$$

$$\sum_{i\in I}\sum_{k\in K_i(w(t))} p_{itks} + \sum_{j\in J} p_{jts} = D_{ts} \qquad \forall t\in T, \forall s\in S \tag{14}$$

$$y_{iwk} \in \{0,1\} \qquad \forall i\in I, \forall k\in K_i, \forall w\in W^o_{ik} \tag{15}$$

All variables other than y_{iwk} are continuous and non-negative. The objective (1) minimizes the sum of cost of the reloading pattern in addition to the sum of the production costs for each scenario and the profit for any remaining fuel for a scenario. Constraints (2) and (3) ensure the reloaded amount in any cycle is always within the possible reloading bounds. Constraints (4) ensure that if a cycle for a plant is set, then all preceding cycles must be set. Constraints (5) model CT21, and all other CT13-C20 constraints are modelled in the form of (6). These typically have a right hand side of one (i.e. give pairwise conflicts), but in some cases can exceed this (in the case of CT19 and CT20). Constraints (7) ensure stock level consistency between the starting stock level of a cycle and its end stock level (taking into account any production), while constraints (8) reflect the requirement that some fuel is lost as a plant goes through a reload. The CT11 constraints are enforced by constraints (9) and constraints (10) respectively, and (11) ensures the stock at the end of the last cycle is the plant's final stock level. Maximum and minimum production required by the respective plants are enforced by (12) and (13). Constraints (14) ensure all demand in each time step is met.

4 Methodology

In this section we present a Benders Decomposition based framework to solve the compact formulation (1)-(15). We begin by providing a short introduction to Benders in general before describing the Benders reformulation of (1)-(15). Once the necessary models have been introduced, we discuss, in detail, the components of the algorithm developed to solve this reformulation.

4.1 Benders Decomposition

Benders Decomposition is a well-known technique for solving large scale mixed integer programming (MIP) problems that have a special block structure [see 1]. It is commonly found in stochastic applications where one is required to make a so-called first stage decision and then, upon the realization of some random event, solve a second problem that ameliorates the first stage decision. This is often the case in the power industry, where the demand is highly stochastic. Recent applications of Benders in the power industry include [see 3, 9, 2, 10]. However, it has also been applied in a variety of other areas including telecommunication network design [see 7], staff scheduling [see 5], aircraft routing and crew planning [see 6], and uncapacitated hub location [see 4].

The Benders approach decomposes the original problem into a mixed integer *master* problem and one or more independent, linear *subproblems*. Consider the following formulation as an example.

$$\begin{aligned} \mu = \min\ & c^T x + f^T y \\ \text{s.t. } & \mathcal{A}x \qquad = b && (16)\\ & \mathcal{B}x + \mathcal{D}y = d && (17)\\ & x \in \mathcal{X} \subseteq \mathbb{R}^p, y \in \mathcal{Y} \subseteq \mathbb{R}^q, \end{aligned}$$

where x and y are vectors of decision variables with dimension p and q, $\mathcal{X}$ and $\mathcal{Y}$ are polyhedrons, $\mathcal{A}$, $\mathcal{B}$, and $\mathcal{D}$ are matrices, and c, f, b, and d are vectors (all with approriate dimensions). The first set of constraints, (16) restrict the values of x, while the second set, (17) restrict the values of both x and y. With Benders Decomposition this problem is decomposed into the following two smaller problems, P1 and P2.

$$\begin{aligned} P1: \min\ & c^T x + z(x) \\ \text{s.t. } & \mathcal{A}x = b \\ & x \in \mathcal{X} \end{aligned} \qquad\qquad \begin{aligned} P2: z(x) = \min\ & f^T y \\ \text{s.t. } & \mathcal{D}y = d - \mathcal{B}x \\ & y \in \mathcal{Y} \end{aligned} \qquad (18)$$

Observe that P1 is an optimization problem in terms of the x variables only, where $z(x)$ is the objective function value of P2 given the solution to P1. If one assumes that P2 is not unbounded, then one can also calculate $z(x)$ by solving it's dual formulation. If u denotes the vector of dual variables associated with constraints (18), then the dual formulation of P2 can be stated as:

$$\begin{aligned} \text{D2: } & \max u^T(d - \mathcal{B}x) \\ & \text{s.t. } D^T u \leq f \end{aligned}$$

The feasible region of this optimization problem is completely independent of the values of x, which only affect the objective function. Assuming that the feasible region of D2 is not empty, then exactly one of two cases will occur when solving D2 for a given solution $\hat{x} \in \mathcal{X}$. Either D2 is unbounded from above, or D2 has a finite optimal solution. In the first case there must exist an extreme ray r_j such that $r_j^T(d - \mathcal{B}\hat{x}) > 0$, while in the second case there must exist an extreme point u_j of the feasible region such that $z(\hat{x}) = u_j^T(d - \mathcal{B}\hat{x})$. If we denote the set of all extreme rays of D2 as R and the set of all extreme points of D2 as U, then D2 can be restated as follows.

$$\begin{aligned} \text{D2}^* \ & \min z \\ & \text{s.t. } (r_i)^T(d - \mathcal{B}x) \leq 0 \ \forall r_i \in R && (19) \\ & \quad (u_i)^T(d - \mathcal{B}x) \leq z \ \forall u_i \in U && (20) \\ & && (21) \end{aligned}$$

P2 now consists of the single variable z. The first set of constraints, (19), restricts the set of solutions to P1 to those which are also feasible for P2 (termed feasiblity cuts), while the second set, (20), restrict the set of solutions to P1 to those that minimize the objective function value of P2 (termed optimality cuts). Hence, the original problem can be restated as:

$$\begin{aligned} \text{RMP: } & \min \ c^T x + z \\ & \text{s.t. } \mathcal{A}x = b \\ & (r_i)^T(d - \mathcal{B}x) \leq 0 \ \forall r_i \in R \\ & (u_i)^T(d - \mathcal{B}x) \leq z \ \forall u_i \in U \\ & x \in \mathcal{X} \end{aligned}$$

Since there can be an exponential number of constraints of the form (19) and (20), it is impractical to generate them all and include them initially. The so-called Restricted Master Problem (RMP) starts with a subset of these and dynamically identifies violated ones as needed. Thus it is an iterative process where at any iteration a candidate solution (x^*, z^*) is found. The subproblem is then solved to calculate $z(x^*)$. If $z(x^*) = z*$, the algorithm terminates, otherwise a violated feasibility or optimality cut exists. One adds the respective cut to the RMP and iterates again. In what follows we provide the Benders reformulation of (1)-(15).

4.2 Benders Reformulation

For the problem under consideration one can observe that once the reload dates and reload amounts have been fixed, one can independently solve each scenario and find the cheapest way of supplying the respective power demand power of each. That is, the problem naturally

decomposes into n independent subproblems, where n is the number of different possible scenarios. Thus, the role of the master problem in this context is to identify good outage/reloading schedule. We model this as a MIP since it contains binary decision variables, which govern reload dates, and continuous variables that reflect the corresponding reload amounts. The Benders RMP (without the addition of any feasibility and optimality cuts) can be stated as follows.

Master problem

$$\min \sum_{i\in I}\sum_{k\in K} c_{ik}r_{ik} + \frac{1}{|S|}\sum_{s\in S}\theta_s \tag{22}$$

$$\text{s.t. } r_{ik} \geq \underline{\mathrm{R}}_{ik} \cdot \sum_{w\in W_{ik}} y_{iwk} \qquad \forall i \in I, \forall k \in K_i \tag{23}$$

$$r_{ik} \leq \overline{R}_{ik} \cdot \sum_{w\in W_{ik}} y_{iwk} \qquad \forall i \in I, \forall k \in K_i \tag{24}$$

$$\sum_{w\in W_{ik}} y_{iwk} \geq \sum_{w\in W_{i,k+1}} y_{i,w,k+1} \qquad \forall i \in I, \forall k \in K_i \tag{25}$$

$$\sum_{i\in C_m}\sum_{k\in K_i}\sum_{w\in IT_m}\sum_{w'=w-L_{ik}+1}^{w} y_{iw'k} \cdot \sum_{t=t(w)}^{t(w+1)-1} P_{it}^{max} \leq I_m^{max} \qquad \forall m \in M_{21}, \forall w \in W \tag{26}$$

$$\sum_{(iwk)\in H} y_{iwk} \leq K_H \qquad \forall H \in \mathcal{H} \tag{27}$$

$$r_{ik} \geq 0 \qquad \forall i \in I, \forall k \in K_i \tag{28}$$

$$y_{iwk} \in \{0,1\} \qquad \forall i \in I, \forall k \in K_i, \forall w \in W_{ik}^o, \tag{29}$$

Associated with each scenario $s \in S$ is a decision variable θ_s that reflects the cost of supplying the power demanded in scenario. The constraints are as described in Section 3. In addition to the constraints described here a number of additional constraints are added to the master problem, which will be described in Section 6. Given a candidate solution (r, y, θ) to this problem one can solve s power production subproblems to separate any violated feasibility and optimality cuts. The structure of the subproblems that must be solved is given below

Subproblem

$$\min \sum_{t\in T}\sum_{j\in J} c_{j,t}F_t p_{j,t} - \sum_{i\in I} c_{i,|T|+1}x^f_i \tag{30}$$

$$\text{s.t. } x^e_{iks} = x^b_{iks} - \sum_{t\in T} p_{itks}\cdot F_t \qquad \forall i\in I, \forall k\in K_i, \forall s\in S \tag{31}$$

$$x^b_{iks} = r_{ik} + B_{ik}\sum_{w\in W^o_{ik}} y_{iwk} + \tilde{Q}_{ik}\left(x^e_{i,k-1,s} - B_{i,k-1}\sum_{w\in W^o_{ik}} y_{iwk}\right) \qquad \forall i\in I, \forall k\in K_i, \forall s\in S \tag{32}$$

$$x^e_{iks} \le A^{max}_{i,k+1} + \left(1 - \sum_{w\in W^o_{ik}} y_{i,w,k+1}\right)\left(M^1_i - A^{max}_{i,k+1}\right) \qquad \forall i\in I, \forall k\in K_i, \forall s\in S \tag{33}$$

$$x^b_{iks} \le S^{max}_{ik} + \left(1 - \sum_{w\in W^o_{ik}} y_{iwk}\right)(M_i - S^{max}_{ik}) \qquad \forall i\in I, \forall k\in K_i, \forall s\in S \tag{34}$$

$$x^f_{is} \le \sum_{k'>k}\sum_{w\in W^o_{ik}} y_{iwk'}M_i + x^e_{iks} \qquad \forall i\in I, \forall k\in K_i, \forall s\in S \tag{35}$$

$$p_{itks} \le \overline{P}_{it}\cdot\rho(i,w(t),k) \qquad \forall i\in I, k\in K_i, t\in T^p_{ik}, \forall s \tag{36}$$

$$\underline{P}_{jts} \le p_{jts} \le \overline{P}_{jts} \qquad \forall j\in J, \forall t\in T, \forall s\in S \tag{37}$$

$$\sum_{i\in I}\sum_{k\in K_i(w(t))} p_{itks} + \sum_{j\in J} p_{jts} = D_{ts} \qquad \forall t\in T, \forall s\in S \tag{38}$$

Again each constraint is as described in section 3. Each subproblem is modelled as a linear program (LP) and determines how much each power plant should produce in each time step so that the demand in each time step for the given scenario is satisfied, and the various constraints regarding fuel stock levels are respected In addition to this one must respect several production level bounds at each power plant. Two such constraints, CT6 and CT12, are two complicated to include in the LP formulation. The first states that once the fuel stock level at a given nuclear power plant falls below a certain threshold production must follow a piecewise linear decreasing function, while the second tries to ensure a high utilization of the nuclear power plants by stipulating that the average deviation of the production cannot be more than a certain tolerance from the maximum possible production level (however, only prior to the aforementioned threshold). As a result, such constraints are not enforced when solving a given subproblem, but rather in a post-processing step that attempts to repair the subproblem solution. This is described in Section 7.1.

In a typical Benders Decomposition fashion, optimality cuts are separated using solutions to each of the subproblems and are added to the master problem to direct it towards more promising outage/reloading schedules. In order to minimize the need for feasibility cuts to the master problem, constraints are preemptively added to the master problem and try to enforce CT11. These constraints also partly enforce CT6 and are discussed in Section 6.

4.3 Solution Approach

In this section we provide an overview of the algorithm we propose for solving the Benders reformulation. Here we simply provide a sketch of the approach, more detailed discussions on certain components of the algorithm are provided in the subsequent sections. The algorithm can be separated into three distinct phases, and we discus each in turn.

Stage 0 In Stage 0 the master problem is solved to integrality without the addition of any optimality cuts. This is done in order obtain a solution that is supposedly easy to repair so that at least one complete solution is obtained. We do not expect this solution to be of high quality, but it is expected to be easy to find. Prior to building the MIP in Stage 0, extensive preprocessing is used to remove redundant variables from the model. This preprocessing step is described in Section 5.1.

Stage 1 In this stage, the root node of the relaxed master problem is solved. The relaxed master problem is obtained by removing the integral restrictions on the y_{iwk} variables. Solving the root node is an iterative procedure between the master problem and the subproblems, where the subproblems are used to separate any violated feasibility and optimality cuts given a solution to the master problem. Note that we do not solve all subproblems per Benders iteration since this would simply take too long. A round robin approach is adopted in which only one subproblem is solved per Benders iteration. Since even solving a single instance of the subproblem can be quite time consuming, an aggregated version is used (see Section 5.2). In the aggregated subproblem, the time step is considered to be weeks as opposed to days or even hours. When no optimality cut has a magnitude of violation greater than some prespecified epsilon, or some predetermined time limit is reached, this stage terminates. Cplex 12.1 is used to solve both the master and the subproblems.

Stage 2 In the final stage of the algorithm the master problem is solved to integrality without the addition of anymore optimality cuts using a standard standard branch-and-bound technique. Cplex's populate routine is used to collect integeral solutions found in the branch-and-bound tree. Once a certain number of integer solutions have been found, all subproblems are solved to obtain a complete solution. However, the complete solution may violate CT6 and CT12. To remedy this, the solution to each subproblem is repaired so that CT6 and CT12 are satisfied. The routine to do this is described in Section 7.1. Once a complete solution satisfying all constraints has been found, a 2-opt heuristic is used to improve its quality. This is detailed in Section 7.2. The best found solution is retained. Stage 2 continues until either all integer solutions from the branch and bound tree are enumerated, or a prespecified time limit is reached. The pseudo code for the complete methodology is given in Algorithm 1.

5 Reducing the problem size

As the problems may contain a huge number of variables, it is an advantage both with respect to computational time and memory consumption to reduce the problem size. In the following we describe two such reduction procedures.

5.1 Preprocessing

For the master problem employed, there is a y_{iwk} variable, for each possible week w the outage of cycle k for plant i can occur. Because many of the constraints (CT13-CT21) concern these outage dates, many of them are infeasible, and removing them in a preprocessing step will reduce the size of the master problem. In the following we present a simple, yet effective preprocessing procedure.

Algorithm 1 Core Methodology

```
Preprocess problem instance
Solve master MIP without any optimality/feasibility cuts
repeat
  Solve relaxed master problem
  Solve next aggregated subproblem
  Separate violated optimality/feasibility cut and append
until No violated optimality/feasibility cuts exist or time limit exceeded
Convert to MIP and run branch-and-bound
repeat
  Populate integral solution pool with a certain number of solutions
  for s ∈ S do
    Solve subproblem associated with scenario s
    Repair subproblem solution
    Run 2-opt heuristic to improve solution quality
  end for
  if A feasible solution is found for each subproblem then
    Update best known solution if total cost is better than that of the current best solution
  end if
until All integer solutions have been enumerated or time limit exceeded
```

Let $G = (V, E)$ be a graph, where each node $v \in V$ corresponds to the outage date, w_v of some cycle, k_v, of plant i_v. There is an edge $(u, v) \in E$, if there is a conflict between the two corresponding outage dates, i.e., it is infeasible for cycle k_u to start its outage in week w_u while cycle k_v starts its outage in week w_v. How the conflicts are derived is explained later. For a set $S \subseteq V$ let $N(S) = \{v \in V \backslash S : \forall u \in S : \exists (u, v) \in E\}$, i.e., the set of nodes incident to all nodes in S. Now if $S \subseteq V$ is a set of nodes, for which it is known that at least one of the corresponding outage dates must be chosen in any solution, then the set $N(S)$ may be removed from the graph, as the corresponding outage dates can never be used. As it is known from the input data that some of the cycles must be scheduled, the set of nodes corresponding to the outage dates of these cycles can be used to perform the above described elimination.

Conflicts between outage dates are derived as follows:

1. All outage dates of the same cycle are in conflict.

2. Assume that the outage of cycle k of plant i occurs in week w, then the outage of any following cycle of the same plant must occur after week $w + L_{ik} - 1$ (see constraint CT13). This can be represented as conflicts between the individual outage dates.

3. Similarly assume that the outage of cycle k of plant i occurs in week w. Because of constraint CT11 the stock must be below $A^{max}_{i,k+1}$ before the outage of the next cycle can occur, and be below $S^{max}_{i,k+1}$ after the reload. Let LB be a lower bound on the stock at the beginning of cycle k, and let $UB(w_0, w_1)$ be an upper bound on the production capacity from week w_0 to week w_1 for plant i. A lower bound on the stock at any time after w, may then be calculated as $LBS(w_1) = LB - UB(w, w_1)$. The outage of cycle $k + 1$ must occur after

$$w_{min} = \arg\min\{w_1 : LBS(w_1) \leq A^{max}_{i,k+1} \wedge LBS(w_1) \leq f(S^{max}_{i,k+1}, \underline{R}_{i,k+1})\}\},$$

where $f(x, r)$ returns the stock after a reload of r given end stock x, as specified by CT10. We set $LB = \underline{R}_{ik}$, and $UB(w_0, w_1)$ is calculated by assuming a production of $\overline{P}_{it}$ as long as the stock is above B_{ik}, and then following the shutdown curve. Again this can be represented as conflicts between the individual outage dates.

4. Constraints CT14-CT18 can be represented as conflicts between individual outage dates.

5. (Optional) The previous methods are exact in the sense that only outage dates which are infeasible are removed. These methods derive a large number of conflicts, and as a consequence a large number of outage dates may be removed. Even so, for some less tightly constrained instances (see section 8 on computational results) this may not reduce the size of the problems enough and we thus include a heuristic for deriving conflicts. The working assumption for this heuristic is that it is not optimal to have a plant type 2 without stock for too long before the next reload occurs. Assume that the outage of cycle k of plant i occurs in week w. Let UB be an upper bound on the stock at the beginning of cycle k, and let $LB(w_0, w_1)$ be a lower bound on the stock which must be consumed from week w_0 to week w_1. Note that this lower bound is not zero because of constraint CT12. Let

$$w_{max} = (1 + \alpha) \arg\min\{w^1 : UB - LB(w, w^1) \leq 0\},$$

where $\alpha \geq 0$. We add conflicts between w and all outage dates $w' > w$ of the following cycle $k + 1$ for the same plant. The value α controls how long we allow a plant to lay idle in the worst case. As UB we use S^{max}_{ik}, and $LB(w_0, w_1)$ is calculated by assuming a production of zero until CT12 is violated, then production at $\overline{P}_{it}$ as long as the stock is above B_{ik}, and then following the shutdown curve.

In addition to the above conflicts we remove certain outage date as follows: Let k be a cycle *that does not necessarily* have to be scheduled. Let $w = T^A + L_{ik}$ be the latest point in time the production campaign of cycle k can start, let w_{max} be defined as above. We remove all outage dates $w' > w$ for the following cycle $k + 1$. This may remove additional outage dates.

Additional conflicts can be deduced by the calculation described in point 3 above if the A^{max}_{ik}, S^{max}_{ik}, or R^{max}_{ik} values can be tightened. For any $i \in I$ and $k \in K_i$ the values may be tightened as follows:

$$\begin{aligned} A^{max}_{ik} &:= \min\{A^{max}_{ik}, \tilde{f}(S^{max}_{ik}, \underline{R}_{ik}\}) \\ S^{max}_{ik} &:= \min\{S^{max}_{ik}, f(A^{max}_{ik}, \overline{R}_{ik})\} \\ \overline{R}_{ik} &:= \min\{\overline{R}_{ik}, \hat{f}(0, S^{max})\} \end{aligned}$$

where $f(x, r)$ is as earlier, $\tilde{f}(y, r)$ gives the end stock which result in stock y after a reload of r as specified by CT10, and $\hat{f}(x, y)$ gives the reload which results in stock y given end stock x, as specified by CT10.

All conflicts of the conflict graph are added to the mater problem as clique constraints, which thus include the constraints CT13–CT18, and only CT19 and CT20 are included using the form (27). The complete preprocessing algorithm is sketched in Algorithm 2.

Algorithm 2 Preprocessing of outage dates

```
Tighten A^max and S^max values.
Construct conflict graph G.
repeat
  for all i ∈ I do
    for all cycles, k, of i which must be scheduled do
      Eliminate vertices of G.
    end for
  end for
until No vertices could be eliminated
```

5.2 Aggregation

Unlike the preprocessing technique described in Section 5.1, which attempts to remove as many redundant variables as possible from the master problem, the aggregation technique focuses on the subproblem and reduces the size of this problem by aggregating the individual time step production variables into variables that determine the weekly production level for each power plant (both type 1 and type 2). Since the time discretization of the master problem is weekly, one does not need the production levels for each individual time step (which can be as short as 4 hours) when solving the subproblem in the cutting phase of our methodology. This simple aggregation approach can dramatically reduce the size of the subproblem; the number of production variables can be reduced by a factor 42 at best. This primarily allows faster Benders iterations to be performed; however, it can also be used to determine the likelihood of finding a feasible solution to the subproblem. If the aggregated version is infeasible, then the disaggregated version will also be infeasible. The reverse, however, is not true. In Section 8 we assess the impact of using the aggregated version in the repair phase of the algorithm. Next, we formalize how both the aggregation and the necessary disaggregation are performed.

Minimal changes are required to model (30)-(38) in order to obtain the aggregated version. In introducing the weekly production variables p_{iwks} and p_{jws}, one is required to update the minimum and maximum production levels for each power plant, i.e. (36) and (37), the demand constraints, i.e. (38), and the production cost for each plant of type 2 to reflect the weekly structure. That is, (36), (37), and (38) become

$$p_{iwks} \leq \rho(i,w,k) \cdot \sum_{t \in w_t} \overline{P}_{it} \quad \forall (i,k,w \in W^p_{ik}, s) \tag{39}$$

$$\sum_{t \in w_t} \underline{P}_{jt} \leq p_{jws} \leq \sum_{t \in w_t} \overline{P}_{jts} \quad \forall (j,w,s) \tag{40}$$

$$\sum_{i \in I} \sum_{k \in K_i(w)} p_{iwks} + \sum_{j \in J} p_{jws} \geq D_{ws}, \quad \forall (w,s) \tag{41}$$

where $D_{ws} = \sum_{t \in w_t} D_{ts}$. The cost of each p_{iwks} variable is assumed to be the average cost of production for the aggregated time intervals, and we assume that the number of time steps per week is constant.

In order to provide a feasible solution to the subproblem, any aggregated solution must be disaggregated (if this is possible). This routine works in a similar way to the repair heuristic

described in Section 7.1. In an aggregated solution one has the weekly production levels of each plant which must be disaggregated into time step production levels in such a way that the demand of each time step is satisfied. Since each type 1 plant has a certain minimum production level in each time step, the procedure begins by first identifying the type 1 plant contribution to the demand in each of the time steps. The respective time step demands are then reduced accordingly. Next, the type 2 power plants are considered in order and an attempt is made to disaggregate their weekly production levels in each of their scheduled cycles. In this disaggregation step one proceeds by assigning the plant's maximum production level in each of the time steps, or the remaining demand for that time step, whichever is the smaller. If disaggregation fails (i.e. the assigned weekly production level for the plant cannot be met), an attempt is made to identify a time step (or as many as required) within the week for which there is unmet demand and for which the plant is currently not producing. If this cannot consume the surplus fuel, one repeats this process but looks across the weeks in the cycle. Finally, an attempt is made to push the remaining fuel to the subsequent cycle as long as CT11 is satisfied. If CT11 is violated, disaggregation is deemed not possible, although there is no guarantee that it is actually not possible. Once disaggregation has been successfully performed for each type 2 power plant, any unmet demand in any time step is satisfied by the cheapest type 1 power plant.

Algorithm 3 Disaggregation Algorithm

Require: A feasible solution to an aggregated subproblem
  for all $j \in J$ **do**
    for all $t \in T$ **do**
      Reduce the demand in time step t by the minimum required production level for plant j.
    end for
  end for
  for all $i \in I$ **do**
    for all cycles k of i that *must* be scheduled **do**
      for all weeks w of k **do**
        Disaggregate weekly production level
        if Surplus power remains **then**
          Try to consume the surplus fuel in the given week. If this is not possible, try to consume the fuel in the given cycle. If the remaining fuel still cannot be used, try to move it to the subsequent cycle. If a CT11 violation occurs, disaggregation is deemed impossible.
        end if
      end for
    end for
  end for

6 Feasibility

The following so-called stock-bounding constraints are added to the master problem in order to decrease the number of infeasible subproblems due to the CT11 constraints. One can think of these constraints as adding an artificial stock variable to the master problem, which

must satisfy these constraints given an upper bound on production. In addition a number of constraints, not described here because of space consideration, are added which makes the complete solution less likely not to be repairable because of CT6, by bounding the amount of fuel which can be consumed between time periods when satisfying the shutdown curve.

Stock Bounding

- x^b_{ik}: lower bound on stock at the beginning of cycle (i,k).
- x^e_{ik}: lower bound on stock at the end of cycle (i,k).

$$x^e_{ik} \geq x^b_{ik} - \sum_{t \in T^p_{ik}} \overline{P}_{it} \cdot D_t \cdot \rho(i, w(t), k) \qquad \forall (i,k) \qquad (42)$$

$$x^b_{ik} = r_{ik} + BO_{ik} \sum_{w \in W_{ik}} y_{iwk} + \frac{Q_{ik} - 1}{Q_{ik}} \left(x^e_{i,k-1} - BO_{i,k-1} \sum_{w \in W_{ik}} y_{iwk} \right) \qquad \forall (i,k) \qquad (43)$$

$$x^e_{ik} \leq A^{max}_{i,k+1} + \left(1 - \sum_{w \in W_{ik}} y_{i,w,k+1} \right) \left(M^1_i - A^{max}_{i,k+1} \right) \qquad \forall (i,k) \qquad (44)$$

$$x^b_{ik} \leq S^{max}_{ik} + \left(1 - \sum_{w \in W_{ik}} y_{iwk} \right) \left(M^1_i - S^{max}_{ik} \right) \qquad \forall (i,k) \qquad (45)$$

Similar to in the subproblem, constraints (42) ensure stock level consistency between the starting stock level of a cycle and its end stock level assuming maximal production in all time steps, while constraints (43) reflect the requirement that some fuel is lost as a plant goes through a reload. The A^{max} bounds and S^{max} bounds are enforced by constraints (44) and constraints (45) respectively.

6.1 Stock Cuts

The Stock Cuts are introduced to enforce some of the structure of the shutdown curve on the stock bound variables $x^b_{ik}, x^e_{ik} : i \in I, k \in K$ defined above. The cuts are divided into three sets described in the following:

Cut-SI

$$x^b_{ik} - \sum_{w' > w} y_{i,w',k+1} \left(UB^1_{ik}(w,w') + A^{max}_{i,k+1} \right) \leq \left(1 - \sum_{w' > w} y_{i,w',k+1} \right) S^{max}_{ik} + (1 - y_{iwk})\, S^{max}_{ik} \quad \forall (i,w,k) \qquad (46)$$

where $UB^1_{ik}(w,w') :=$ max production from w to w' assuming S^{max}_{ik} at time w with no intermediate refueling. $UB^1_{ik}(w,w')$ is bounded from above by S^{max}_{ik}.

There are three parts to these cuts:

- $y_{iwk} = 0$: This means that for plant i, week w is not the date of outtage for cycle k. In this case the cut evaluates to $x^b_{ik} \leq S^{max}_{ik} + \rho$ with ρ being some positive number. This does not bound x^b_{ik} further than the allready existing bound of S^{max}_{ik}.
- $y_{iwk} = 1$ and $\sum_{w' > w} y_{i,w',k+1} = 0$: This means that for plant i, week w is the date of outtage for cycle k and for cycle $k+1$ there is no outtage. In this case the cut evaluates to $x^b_{ik} \leq S^{max}_{ik}$, which does not bound x^b_{ik} further.

- $y_{iwk} = 1$ and $y_{i,w',k+1} = 1$ for some $w' > w$: This means that for plant i, week w is the date of outtage for cycle k and for the following cycle $k+1$ week w' is the date of outtage. In this case the cut evaluates to $x^b_{ik} - UB^1_{ik}(w,w') \leq A^{max}_{i,k+1}$, which ensures that the begin stock x^b_{ik} of cycle k is small enough for the maximum permitted fuel prior to reload in cycle $k+1$ is not violated, assuming maximum production in cycle k and no interactions from other plants $j \in I : i \neq j$.

Special case. For $k = 0$:

$$XI - \sum_{w'>w} y_{i,w',k+1} \left(UB^1_{ik}(w,w') + A^{max}_{i,k+1}\right) \leq \left(1 - \sum_{w'>w} y_{i,w',k+1}\right) XI \quad \forall (i,0,0) \tag{47}$$

Similar to above several cases exist:

- $\sum_{w'>w} y_{i,w',k+1} = 0$: Evaluates to $XI \leq XI$, which requires no further comments.
- $y_{i,w',k+1} = 1$ for some $w' > w$: Evaluetes to $XI - UB^1_{ik}(w,w') \leq A^{max}_{i,k+1}$. By following the argumentation above, this can be shown to be valid.

Cut-SII

$$\underline{R}_{ik} - \sum_{w'>w} y_{i,w',k+1} UB^2_{ik}(w,w') \leq x^e_{ik} + \left(1 - \sum_{w'>w} y_{i,w',k+1}\right) \underline{R}_{ik} + (1 - y_{iwk})\, \underline{R}_{ik} \quad \forall (i,w,k) \tag{48}$$

where $UB^2_{ik}(w,w') :=$ max production from w to w' assuming $\underline{R}_{i,k}$ at time w.

As for *Cut-SI* there are three parts to these cuts:

- $y_{iwk} = 0$: In this case the cut evaluates to $0 \leq x^e_{ik} + \rho$ with ρ being some positive number. This does not bound x^b_{ik} further than the allready existing bound of 0.
- $y_{iwk} = 1$ and $\sum_{w'>w} y_{i,w',k+1} = 0$: In this case the cut evaluates to $0 \leq x^e_{ik}$, which does not bound x^b_{ik} further.
- $y_{iwk} = 1$ and $y_{i,w',k+1} = 1$ for some $w' > w$: In this case the cut evaluates to $\underline{R}_{ik} - UB^2_{ik}(w,w') \leq x^e_{ik}$, which ensures that the begin stock x^b_{ik} is large enough compared to the minimum fuel reload, assuming maximum production in cycle k and no interactions from other plants $j \in I : i \neq j$.

Special case. For $k = 0$:

$$XI - \sum_{w'>w} y_{i,w',k+1} UB^2_{ik}(w,w') \leq x^e_{ik} + \left(1 - \sum_{w'>w} y_{i,w',k+1}\right) XI \quad \forall (i,0,0) \tag{49}$$

Similar to above several cases exist:

- $\sum_{w'>w} y_{i,w',k+1} = 0$: Evaluates to $0 \leq x^e_{ik}$, which requires no further comments.
- $y_{i,w',k+1} = 1$ for some $w' > w$: Evaluetes to $XI - UB^2_{ik}(w,w') \leq x^e_{ik}$. By following the argumentation above, this can be shown to be valid.

Cut-SIII

$$x^b_{ik} - UB^1_{ik}(w,w') \leq x^e_{ik} + (2 - y_{iwk} - y_{i,w',k+1})\, S^{max}_{ik} \quad \forall i \in I, \forall k \in K, \forall w \in W_{ik}, \forall w' \in W_{i,k+1}, w < w' \tag{50}$$

where $UB^1_{ik}(w,w')$ is defined as before.

There are two parts to these cuts:

- $y_{iwk} + y_{i,w',k+1} \leq 1$: In this case the cut evaluates to $x^b_{ik} \leq x^e_{ik} + \rho$ with ρ being some positive number, since $UB^1_{ik}(w, w') \leq S^{max}_{ik}$. This is clearly dominated by the constraint $x^b_{ik} \leq x^e_{ik}$.
- $y_{iwk} = 1$ and $y_{i,w',k+1} = 1$: In this case the cut evaluates to $x^b_{ik} - UB^1_{ik}(w, w') \leq x^e_{ik}$, which ensures that the end lower bound on stock x^e_{ik} compared to the start lower bound on stock is not smaller than what can be explained by a maximum production in the cycle.

7 Postprocessing

The role of the postprocessing stage is to try to convert a solution, in the following also referred to as the reference solution, which does not satisfy the CT6 and CT12 constraints into one that does. This process is divided in two stages: in the first stage, called the repair stage, the production levels and reload amounts are altered in an attempt to satisfy CT6, and CT12, without violating any other constraints. If the solution could not be repaired, it is discarded. In the second stage, called the postoptimization stage, the production levels are shuffled between plants in an attempt to reduce the cost of the solution. The two stages are now described in further details.

7.1 Repair

The input to this stage is a solution that satisfies all constraints except perhaps CT6, and CT12, i.e., the production curve may not follow the shutdown curve it should, or the maximum modulation is exceeded. The assumption is that the structure of the reference solution is good, and by making small adjustments, it is possible to make it satisfy these two additional constraints without changing the cost too much. Thus we want alterations to be as local as possible and since changes in start and end stock of a cycle propagates to the remaining cycles, the changes in these should be as small as possible. Satisfying CT6 means reducing production levels in some places, while satisfying CT12 means increasing production in some places. Changing the production levels from the reference solution, means that the stock levels passed from one cycle to the next will change from the reference solution. One observes that these changes in stocks can be kept small if a decrease of production in one time step of a cycle can be absorbed by an increase in production in another place of the cycle (see Figure 2 for an example). Changes in production within a cycle could also be absorbed by a change of the amount of fuel reloaded, but this is at odds with the principle of locality, as all scenarios are affected and previous repaired scenarios would thus have to be repaired again.

We now describe the consequences on the remaining constraints, when the production levels are changed: Lowering the production will raise the end stock of the cycle, which may lead to CT11 becoming violated, either for that or a later cycle. Raising production will lower the end stock, which may lead to shortage of fuel in later cycles, where demand can no longer be met.

The repair procedure is divided in two stages: In stage 1 only Type 2 plants are considered, and the production curves of these are adjusted so that they satisfy all constraints except perhaps the demand constraints. Then in stage 2, the production of Type 1 plants are adjusted such that demand is covered. If any of the stages fail, the entire reference solution is discarded. We now describe these two stages in detail.

Stage 1 For each plant $i \in I$, each cycle $k \in K_i$ is treated one at a time, starting with the earliest. Each time a cycle is repaired one of two cases may happen:

1. No change in the stock levels at the start or end occurred. This means all changes of production levels within the cycle, was absorbed by increasing of decreasing production somewhere else within that cycle.
2. Given the repaired production levels, the end stock would be increased by δ. In this case the algorithm has two possibilities: either backtrack and try to have δ less stock at the beginning, i.e., consume δ more earlier, or push the stock excess to the next cycle. The algorithm first backtracks, and if this is not possible pushes the stock to the next cycle.

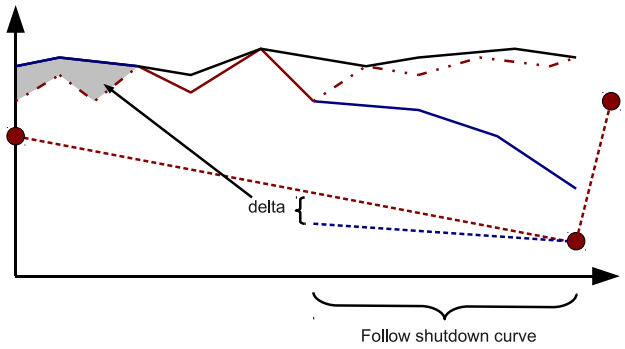


Figure 2: Example of repairing a cycle, such that the shutdown curve is respected by shuffling production to an earlier part of the cycle, such that the end stock remains the same. The upper black line is the max production capacity, the upper red dashed line is the production levels before repair, the upper blue line is the repaired shutdown curve, the upper full red line is the production levels unaltered by the repair. The lower red line is the stock levels in the reference solution, while the lower blue line is the stock levels assuming the shutdown curve is followed backwards from the end stock. δ is the extra stock that must be consumed earlier in the cycle for the end stock to remain the same. The gray area represents the increase in production in order to consume the extra stock.

The repair algorithm is sketched in Algorithm 4, where x^b_{ik} and x^e_{ik} is the stock at the beginning and end of the cycle respectively, t^b_{ik} and t^e_{ik} is the beginning and end respectively of the production campaign of that cycle, and $\delta_{ik} \in \mathrm{R}$ is the amount of stock to clear, either from a backtrack from a later cycle, or from an earlier cycle.

Stage 2 This stage is quite simple. For each $t \in T$ it is checked whether demand is either over-supplied or under-supplied. If demand is oversupplied the production of the most expensive Type 1 plants are reduced, if demand is under-supplied the production of the least expensive Type 1 plants are raised. It may happen that demand can not be meet because of the bounds on the production of plant Type 1. If so production is attempted shuffled within each cycle in a manner similar to the produce described in the next section. If demand can not be meet, the reference solution is discarded.

7.2 Postoptimization

The input to this stage is a solution that satisfies all constraints, and the role of the postoptimization is to try and reduce the cost of the solution by performing alterations, which do not lead to any new constraint violations but reduces the overall cost. As performing alterations which result in changes to the end stock of a cycle propagates, calculating the consequence of such alterations can be cumbersome, and we thus restrict our attention to alterations, where this is not the case.

One such alteration is the following: Let t_1 and t_2 be two points in time lying before the start of the shutdown curve within the same cycle for some $i \in I$, let $j_1, j_2 \in J$ be two plants such that $c_{j_1t_1} < c_{j_2t_2}$, and let $\delta = \min\{\overline{P}_{it_2} - p_{it_2}, p_{it_1}, \overline{P}_{j_1t_1} - p_{j_1,t_1},\ p_{j_2,t_2} - \underline{P}_{j_2t_2}\}$, where p is the current production level. Then updating $p_{it_1} := p_{it_1} - \delta, p_{it_2} := p_{it_2} + \delta, p_{j_1t_1} := p_{j_1t_1} + \delta, p_{it_2} := p_{it_2} - \delta$, results in an improved cost while satisfying all constraints and not altering the end stock, nor the shutdowncurve of the cycle in question. The postoptimization heuristic is sketched in Algorithm 5.

Algorithm 4 Repair algorithm for a single cycle

Require: A plant $i \in I$ and cycle $k \in K_i$
  if is_backtrack **then**
    $x^e_{ik} := \max\{0, x^e_{ik} - \delta_{ik}\}$.
  else
    $x^s_{ik} := x^e_{ik} + \delta_{ik}$.
  end if
  Calculate shutdown curve backwards from x^e_{ik}. Let t_B and x_B be the resulting time step and stock right before entering this shutdown curve, and let x_{t_B} be the current stock at time t_B.
  Set $\delta := x_B - x_{t_B}$.
  Raise production by δ (if possible) in the time interval $[t_{ik}; t_B]$. Let δ be what is left.
  if $\delta = 0$ **then**
    Check if CT12 is violated, if so augment production. This may change the point of the shutdown curve and the end stock. Let x be the (new) end stock.
    $\delta_{i,k+1} := x - x^e_{ik}$
    Set is_backtrack := **false**.
    Set $x^e_{ik} := x$.
    Proceed with next cycle.
  else
    Since all production is at the upper bound, CT12 is satisfied.
    if $k = 0$ **then**
      Set can_backtrack := **false**
    end if
    if can_backtrack **then**
      Set $\delta_{i,k-1} := \delta$
      Backtrack to previous cycle.
    else
      Set $\delta_{i,k+1} := \delta$.
      Set $x^e_{ik} := x^e_{ik} + \delta$.
      Proceed with next cycle.
    end if
  end if

8 Computational results

In this section we present the computational experiments performed. The challenge instances are divided in three groups: data0–data5 are the initial instances used for the qualification phase, data6–data10 are the instances made public after the qualification phase, finally data11–data15 are the instances used for the final ranking of the competitors. These instances were not made available until after the end of the challenge. As only data instances data0–data10 were available, we restrict the experiments to these 11 instances, and consider only all the instances for the final computational results. For some experiments we further restrict our attention to a representative sample: data1, data5, data7, data8, and data10.

Table 2 lists the characteristics of the instance, where $|T|$ is the number of time steps, $|W|$ is the number of weeks, $|K|$ is the number of cycles, $|S|$ is the number of scenarios, $|J|$ is the number of plants of type 1, $|I|$ is the number of plants of type 2, and **13–21** are the number of corresponding CTXX constraints.

Algorithm 5 Postoptimization heuristic

Require: A solution satisfying all constraints.
  for all $i \in I$ **do**
    for all cycles, k, of i **do**
      for some number of iterations **do**
        Select at random some t_1, t_2 lying within k and before the start of a shutdown curve.
        Select at random some $j_1, j_2 \in J$ such that $c_{j_1 t_1} < c_{j_2 t_2}$.
        $\delta = \min\{\overline{P}_{it_2} - p_{it_2}, p_{it_1}, \overline{P}_{j_1 t_1} - p_{j_1,t_1}, p_{j_2,t_2} - \underline{P}_{j_2 t_2}\}$,
        Update $p_{it_1} := p_{it_1} - \delta, p_{it_2} := p_{it_2} + \delta, p_{j_1 t_1} := p_{j_1 t_1} + \delta, p_{it_2} := p_{it_2} - \delta$
      end for
    end for
  end for

Table 2: Characteristics of the problem instances

Name	$\|T\|$	$\|W\|$	$\|K\|$	$\|S\|$	$\|J\|$	$\|I\|$	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**	**21**
data0	623	89	2	2	1	2	4	1	0	0	0	0	0	0	0
data1	1750	250	6	10	11	10	46	7	0	1	3	0	1	1	1
data2	1750	250	6	20	21	18	84	13	0	1	3	0	1	1	1
data3	1750	250	6	20	21	18	80	10	2	1	3	2	1	1	1
data4	1750	250	6	30	31	30	122	19	0	1	3	0	1	1	1
data5	1750	250	6	30	31	28	120	18	0	1	3	0	1	1	3
data6	5817	277	6	50	25	50	222	33	40	1	3	0	1	50	5
data7	5565	265	6	50	27	48	192	31	35	1	3	0	1	50	5
data8	5817	277	6	121	19	56	114	37	45	1	3	0	1	50	5
data9	5817	277	6	121	19	56	114	37	45	1	3	0	1	50	5
data10	5565	265	6	121	19	56	235	37	45	1	3	0	1	50	5
data11	5817	277	6	50	25	50	239	33	40	1	3	0	1	50	5
data12	5523	263	6	50	27	48	207	31	35	1	3	0	1	50	5
data13	5817	277	6	121	19	56	260	37	45	1	3	0	1	50	5
data14	5817	277	6	121	19	56	256	37	45	1	3	0	1	50	5
data15	5523	263	6	121	19	56	245	37	45	1	3	0	1	50	5

Setup The computational experiments were performed on a machine with 2 Intel(R) Xeon(R) CPU X5550 @ 2.67GHz (16 cores in total), with 24 GB of RAM, and running Ubuntu 10.4. The version of CPLEX used is 12.1.

Preprocessing We here examine the effect of the preprocessing described in Section 5.1. Table 3 shows the results: It gives the total number of possible outage dates before preprocessing (**Total**), and

the percentage of these removed by the preprocessing (**Rem.**). As can be seen the preprocessing is very effective removing $80\% - 90\%$ of the possible outage dates for all instances except the very small instance data0, and data8 and data9. For the two latter the number of variables is around twice as large as the larges of the other instances, and less variables are removed (around 63% and 68% respectively). The reason can be gleamed from Table 2: data8 and data9 has much fewer CT13 constraints, i.e., few cycles *must* be scheduled which means the problem is less constrained and eliminating outage dates is harder.

Table 3: Number of variables removed by preprocessing

Name	**Total**	**Rem.**	**Name**	**Total**	**Rem.**
data0	36	28.78%	data6	24683	85.65%
data1	3920	87.53%	data7	35817	80.61%
data2	7941	88.47%	data8	69481	68.03%
data3	8207	89.83%	data9	69136	62.70%
data4	17514	89.41%	data10	30061	85.43%
data5	15415	82.13%			

Heuristic preprocessing As we saw earlier the preprocessing is very effective, but it is still lacks on certain large instances (data8 and data9), where few cycles *has* to be scheduled. For this reason the heuristic conflict detector described as point 5 in Section 5.1 is included. As described there the value of α controls the aggressiveness of the heuristic (lower values means more conflicts). We here examine the effect of including the heuristic preprocessing. The value of α is fixed to 0.05 (for smaller values some instances were infeasible). Table 4 shows the percentage of variables removed (**Rem.**) and the final solution (**Sol.**), without the heuristic (**No Heur.**), and with the heuristic (**Heur.**) given 3600 seconds respectively. For this experiment the stock constraints were added as described for Run 6 in Table 6, aggregation was enabled and postoptimization was disabled. As can be seen, the effect on the final solution quality is minor for data1–data3, for data4–data5 the solution is improved, while it is worsened for data6, data7, and data10, finally we are now able to solve data8, which was not possible earlier.

Aggregation Next we examine the effect of aggregation on the solution quality. To that effect the algorithm is run for 3600 seconds with aggregation enabled and disabled for stage 2 (aggregation is always performed for stage 1). Table 5 shows the results, where **Vars.** is the number of variables in the subproblem, **Cons.** is the number of constraints in the subproblem, and **Sol.** is the final solution. For this experiment the stock constraints were added as described for Run 7 in Table 6, heuristic conflicts were included with $\alpha = 0.05$ and postoptimization was disabled. As can be seen the aggregation results in a big reduction in the number of variables and constraints of the subproblem. For data8, and data10 no solution is found without the use of aggregation.

Stock constraints We here examine the effect of including the constraints described in Section 6, in an attempt to ensure the feasibility of the subproblem. One can choose to include either all the constraints, or only a subset, and to included them only in stage 2 or in both stages. Nine runs are performed, with the settings described in Table 6. The results can be seen in Table 7. For this experiment heuristic conflicts were included with $\alpha = 0.05$, aggregation was enabled and postoptimization was disabled. As can be seen only Run 7 and Run 10 completes for all the tested instances. Run 10 achieves the best average results.

Table 4: Effect of using heuristic preprocessing for different values of α

	No Heur.		Heur.	
Name	**Rem.**	**Sol.**	**Rem.**	**Sol.**
data0	28.78%	**8.7371e12**	28.78%	**8.7371e12**
data1	87.53%	1.6990e11	87.63%	**1.6971e11**
data2	88.47%	**1.4654e11**	88.64%	1.4672e11
data3	89.83%	**1.5537e11**	89.92%	1.5578e11
data4	89.41%	1.1342e11	89.64%	**1.1309e11**
data5	82.13%	1.3272e11	83.28%	**1.3153e11**
data6	85.65%	**9.0945e10**	85.72%	9.2508e10
data7	80.61%	**1.2307e11**	80.68%	1.3663e11
data8	68.03%	–	77.15%	**3.2392e12**
data9	62.70%	–	74.13%	–
data10	85.43%	**1.5303e11**	85.44%	1.7455e11

Table 5: Effect on final solution of aggregating versus not aggregating in stage 2

	Enabled			Disabled		
Name	**Vars.**	**Cons.**	**Sol.**	**Vars.**	**Cons.**	**Sol.**
data1	5514	8693	1.**6971e11**	37692	58871	1.6968e11
data5	16696	25199	1.3147e11	114346	170849	**1.3100e11**
data7	25898	34181	1.3663e11	529438	686121	**1.3412e11**
data8	41762	48309	**3.2392e12**	860182	977529	–
data10	23150	29457	**1.7455e11**	469330	581637	–

Postoptimization We here examine the effect of the postoptimization procedure described in Section 7.2. For each of the 11 instances three runs are performed, with the number of iterations respectively set to $50,000$, $150,000$, and $300,000$. The results can be seen in Table 8. For this experiment the stock constraints were added as described for Run 10 in Table 6, heuristic conflicts were included with $\alpha = 0.05$. The reason for the factor 10 increase in solution quality for data8 over previous results, is a newer version of the code, where the conflict graph more effectively is added as cliques. Due to time constraints the previous tests could not be rerun. As can be seen there is a clear correlation between the number of postoptimization iterations and the final solution quality.

Time We finally examine the solution quality as a function of total time given to the algorithm. Each of the 16 instance is run for respectively 3600 seconds, and 10800 seconds. The results can be seen in Table 9, and Table 10 respectively, where the number in parenthesis is the deviation from the best known

Table 6: Description of the different settings used for the stock constraint runs

Run	Description
1	No stock constraints included
2	Stock constraints (42)–(45) included in stage 1.
3	Stock constraints (42)–(45), SI and SII included in stage 1.
4	All stock constraints included in stage 1.
5	Stock constraints (42)–(45) included in stage 2.
6	Stock constraints (42)–(45), SI and SII included in stage 2.
7	All stock constraints included in stage 2.
8	Stock constraints (42)–(45) included in stage 1, SI and SII included in stage 2.
9	Stock constraints (42)–(45) included in stage 1, remaining included in stage 2.
10	Stock constraints (42)–(45), SI and SII included in stage 1, remaining included in stage 2.

solutions reported on the ROADEF/EURO 2010 challenge website (`http://challenge.roadef.org/2010`). For the challenge a maximum time of 1800 seconds was allowed for the first 6 instances, and 3600 seconds for the remaining 10. Each table gives the following information: the number of optimality cuts added (**#Cuts**), the number of solutions found in stage 2(**#Sols**), the number of solutions found in stage 2 that were repairable (**#Rep**), the solution value (**Sol.**), the percentage deviation from the best known solution (**#Dev**), and the average deviation for the two test sets of five instances (**#Avg**).

As can be seen from the tables, the solution approach performs satisfactorily on instances zero to five. These were the test instances used in the qualification phase of the contest and are less complicated than the second and third set of instances (data6 to data10, and data11 to data15). For the latter sets, the algorithm runs into difficulty due to the large number of binary variables, particularly for data8 and data9 which are far from the best known solutions, and for data13 which is not solved at all. Furthermore, formulating and solving the subproblem as an LP and repairing its solution so that it satisfies CT6 and CT12 appears to be an expensive process, despite the aggregation.

For the the smaller instances, many of the solutions are repairable, while for the larger instances, there is a lot more variation. It is suprising that for two of the instances where the algorithm performs poorly (data8 and data13), there is a large number of solutions found but only a single one is repairable in one case, while none in the other. Generally it appears that for the larger instances, either the solutions to the Master problem can not be adjusted such that they satisfy CT6 and CT12, or the repair algorithm does a poor job.

Doubling the amount of time (Table 10) does not significantly change the results and only data1, data5, and data6 are improved. The trend of finding many solutions which are non-repairable remain the same.

Table 7: Effect of including stock constraints. See Table 6 for a description of each run.

Name	**Run 1**	**Run 2**	**Run 3**	**Run 4**	**Run 5**
data 1	1.6986e11	1.6987e11	1.6981e11	1.6986e11	1.6973e11
data 5	–	1.2656e11	**1.2593e11**	1.2707e11	1.3163e11
data 7	–	1.3535e11	**1.0514e11**	1.2517e11	1.3363e11
data 8	–	–	–	2.8047e12	–
data10	–	1.3055e11	1.3785e11	1.3667e11	**1.1532e11**
Avg.	–	–	–		–

Name	**Run 6**	**Run 7**	**Run 8**	**Run 9**	**Run 10**
data 1	**1.6972e11**	**1.6972e11**	1.6987e11	1.6979e11	1.6977e11
data 5	1.3039e11	1.3148e11	1.2656e11	1.2654e11	**1.2593e11**
data 7	1.3392e11	1.3663e11	1.4035e11	1.3373e11	1.0563e11
data 8	–	3.2393e12	–	–	**2.1963e12**
data10	1.5162e11	1.7455e11	1.4273e11	1.6093e11	1.3864e11
Avg.	–	7.7034e11	–	–	5.4725e11

Table 8: Effect of postoptimization procedure. The number of iterations for the runs are respectively $50,000$, $150,000$, and $300,000$

Name	**Run 1**	**Run 2**	**Run 3**
data1	1.69711e11	1.69710e11	**1.69709e11**
data5	1.26452e11	1.26453e11	**1.26451e11**
data7	1.10602e11	1.09518e11	**1.09013e11**
data8	2.88400e11	2.79264e11	**2.75348e11**
data10	1.30261e11	1.28788e11	**1.28443e11**

9 Conclusion

In conclusion, we have developed a Benders Decomposition approach to solve the large scale energy management problem posed for the ROADEF/EURO 2010 challenge. The approach includes a MIP model of the problem along with additional constraints for ensuring feasibility of the subproblems, a very effective preprocessing and aggregation scheme, which reduces the size of the problem significantly, and an algorithm for repairing a solution which only satisfies a subset of the constraints.

On the first set of instances the approach is competitive, while on the the second two set of instances it is not. This is mainly due to the size of the problems, and the time allotted. On the second set of instances and 5 blind instances we placed 14th out of 19 teams in the final of the competition. One of the few optimal methods proposed, it was unable to compete with the heuristics given only 3600

Table 9: Results for different problem instances given 3600 seconds

Name	**#Cuts**	**#Sols.**	**#Rep.**	**Sol.**	**Dev.**	**Avg.**
data0	5	2517	2517	8.7372e12	0.0709%	
data1	6	352	312	1.6971e11	0.1008%	
data2	17	82	82	1.4629e11	0.1639%	
data3	14	86	86	1.5475e11	0.2050%	
data4	23	36	35	1.1206e11	0.4157%	
data5	21	38	37	1.2645e11	0.4997%	0.2427 %
data6	14	12	12	9.0113e10	8.0173%	
data7	5	121	1	1.0901e11	34.2953%	
data8	3	1486	2	2.7535e11	236.0938%	
data9	9	7	3	3.5103e12	4193.8891%	
data10	37	7	5	1.2844e11	62.3461%	906.9283 %
data11	61	447	11	8.8464e10	14.0143%	
data12	20	12	12	8.8135e10	15.2850%	
data13	16	1017	0	–	–%	
data14	10	8	4	1.0092e11	32.4820%	
data15	46	35	2	1.5758e11	109.8220%	42.9008%

seconds of computing time. The sophisticated approach can, however, provide information as to the quality of solutions through the lower bound information which can be obtained at each iteration of the Benders algorithm aswell as insights into the structire on the problem.

Table 10: Results for different problem instances given 10800 seconds

Name	#Cuts	#Sols.	#Rep.	Sol.	Dev.	Avg.
data0	5	2517	2517	8.7372e12	0.0709%	
data1	6	872	583	1.6971e11	0.1007%	
data2	17	153	153	1.4629e11	0.1639%	
data3	14	165	165	1.5475e11	0.2050%	
data4	23	73	72	1.1206e11	0.4156%	
data5	21	76	74	1.2643e11	0.4842%	0.2401 %
data6	14	23	23	8.9659e10	7.4733%	
data7	5	481	22	1.0901e11	34.2953%	
data8	3	4292	2	2.7535e11	236.0938%	
data9	9	12	10	3.5103e12	4193.8891%	
data10	37	10	9	1.2844e11	62.3461%	906.8179 %
data11	61	458	22	8.8464e10	14.0143%	
data12	20	25	25	8.8135e10	15.2850%	
data13	16	2187	0	–	–%	
data14	10	118	8	1.0092e11	32.4820%	
data15	46	126	2	1.5758e11	109.8220%	42.9008%

References

[1] Jacques F. Benders. Partitioning procedures for solving mixed variables programming problems. *Numerische Mathematik*, 4:238 – 252, 1962.

[2] Jordi Cabero, Mariano J. Ventosa, Santiago Cerisola, and Álvaro Baíllo. Modeling risk management in oligopolistic electricity markets: A benders decomposition approach. *IEEE Transactions on power systems*, 25(1):263 – 271, 2010.

[3] Salvador Perez Canto. Application of benders decomposition to power plant preventive maintenance scheduling. *European Journal of Operational Research*, 184:759 – 777, 2008.

[4] Ivan Contreras, Jean-François Cordeau, and Gilbert Laporte. Benders decomposition for large scale uncapacitated hub location. Technical Report CIRRELT-2010-26, Interuniversity research centre on enterprise networks, logistics, and transportation, 2010.

[5] Olivier Guyon, P Lemaire, Eric Pinson, and David Rivreau. Cut generation for an integrated employee timetabling and production scheduling problem. *European Journal of Operational Research*, 201:557 – 567, 2010.

[6] Anne Mercier, Jean-François Cordeau, and François Soumis. A computational study of benders decomposition for the integrated aircraft routing and crew scheduling problem. *Computers and Operations Research*, 32:1451 – 1476, 2005.

[7] Joe Naoum-Sawaya and Samir Elhedhli. A nested benders decomposition approach for telecommunication network planning. *Naval Research Logistics*, 57:519 – 539, 2010.

[8] Marc Porcheron, AgnÃ¨s Gorge, Olivier Juan, Tomas Simovic, and Guillaume Dereu. Challenge roadef/euro 2010 : A large-scale energy management problem with varied constraints. http://challenge.roadef.org/2010/sujetEDFv22.pdf, 2009.

[9] T. N. Santos and A. L. Diniz. Feasibility and optimality cuts for the multi-stage benders decomposition approach: Application to the network constrained hydrothermal scheduling. In *Power & Energy Society General Meeting, 2009. PES '09. IEEE*, 2009.

[10] Lei Wu and Mohammad Shahidehpour. Accelerating the benders decomposition for network-constrained unit commitment problems. *Energy Systems*, 1:339 – 376, 2010.

This thesis presents how to parallelize a shortest path labeling algorithm. It is shown how to handle Chvátal-Gomory rank-1 cuts in a column generation context. A Branch-and-Cut algorithm is given for the Elementary Shortest Paths Problem with Capacity Constraint. A reformulation of the Vehicle Routing Problem based on partial paths is presented. Finally, a practical application of finding shortest paths in the telecommunication industry is shown.

DTU Management Engineering
Department of Management Engineering
Technical University of Denmark

Produktionstorvet
Building 424
DK-2800 Kongens Lyngby
Denmark
Tel. + 45 45 25 48 00
Fax +45 45 93 34 35

www.man.dtu.dk